CropWatch-RDSIF

Real-time delivery of spatial-temporal information for farmers.

User Manual (document)

UNE COSC591/594 (Group A & C)

COSC594 students:
Jason O. Aboh
Steve Daniel

COSC591 students:
Zoe Dowsett
Qiaoling (Ling) Chen

Table of contents

Project team and Client	**2**
Stakeholders	**2**
User Manual	**3 - 48**
CropWatch-RDSIF: Cloud Function to obtain farm-calculations data	**7 - 25**
CropWatch-RDSIF: Exporting farm data to the Database (BigQuery)	**26 - 30**
CropWatch-RDSIF: Visualizing Database information in DataStudio	**31 - 42**
Repository (Github link)	**42**
Notices	**43**
Licenses	**44 - 48**

Project:

CropWatch: Real-time Delivery of Spatial-temporal Information for Farmers.

CropWatch-RDSIF is an application built to deliver interactive maps to farmers, showing details of their crops over time. The data for the maps are time-series data obtained from satellite and other spatio-temporal sources. The application gathers the data, filters out important information bands within the data, then exports the data into a cloud database which conducts transformations and calculations on that data to provide useful information to farmers via data visualisation reports.

Project team and client

UNE COSC591 students:

- Zoe Dowsett
- Qiaoling (Ling) Chen

UNE COSC594 students:

- Jason O. Aboh
- Steve Daniel

Client:

- James Brinkhoff from Applied Agricultural Remote Sensing Centre (AARSC)

Stakeholders

- Project team: Zoe Dowsett, Qiaoling (Ling) Chen, Jason O. Aboh, and Steve Daniel
- Client: James Brinkhoff (AARSC)
- Project sponsor/Supervisor: Edmund Sadgrove (UNE)
- Suppliers: Google Cloud Platform, satellite and other spatio-temporal sources.
- End users: Farmers in Australia

CropWatch

Real-time delivery of spatial-temporal information for farmers

User Manual

[page left blank on purpose]

CropWatch-RDSIF

Real-time delivery of spatial-temporal information for farmers

User manual

By: Jason O. Aboh

Release 0.1

Before using this information, please be sure to read the general information contained under the “Notices” on page 42 and “Licenses” chapter on page 43 to page 47.

This edition applies to

CropWatch-RDSIF: Cloud Function to obtain farm-calculations data

The python code to create the CropWatch-RDSIF data is hosted within a Google Cloud Platform function using a Python 3.9 runtime. The function is named 'farm-calculator' and consists of 4 files: "main.py", "requirements.txt", "farm_details.json" , and "sa-private-key.json. These files are required to run the cloud function successfully.

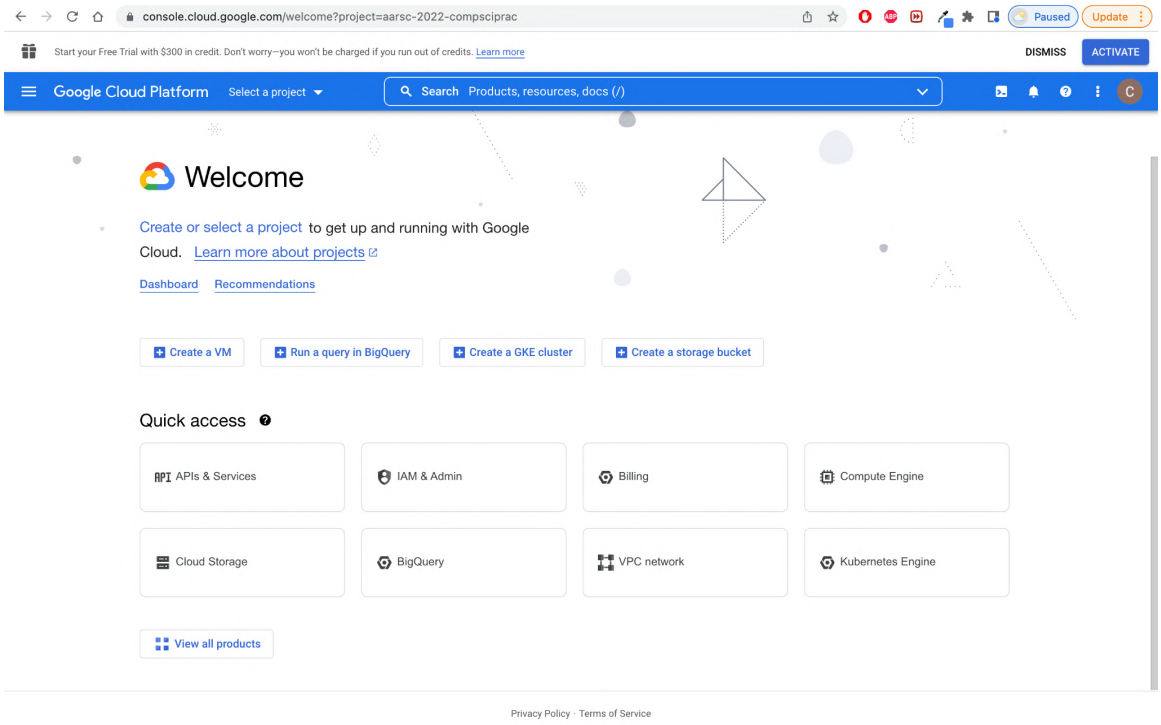
STEPS:

Getting to the project:

- After project set up and an appropriate assignment of permissions and access to team members; Open up your browser and head to :

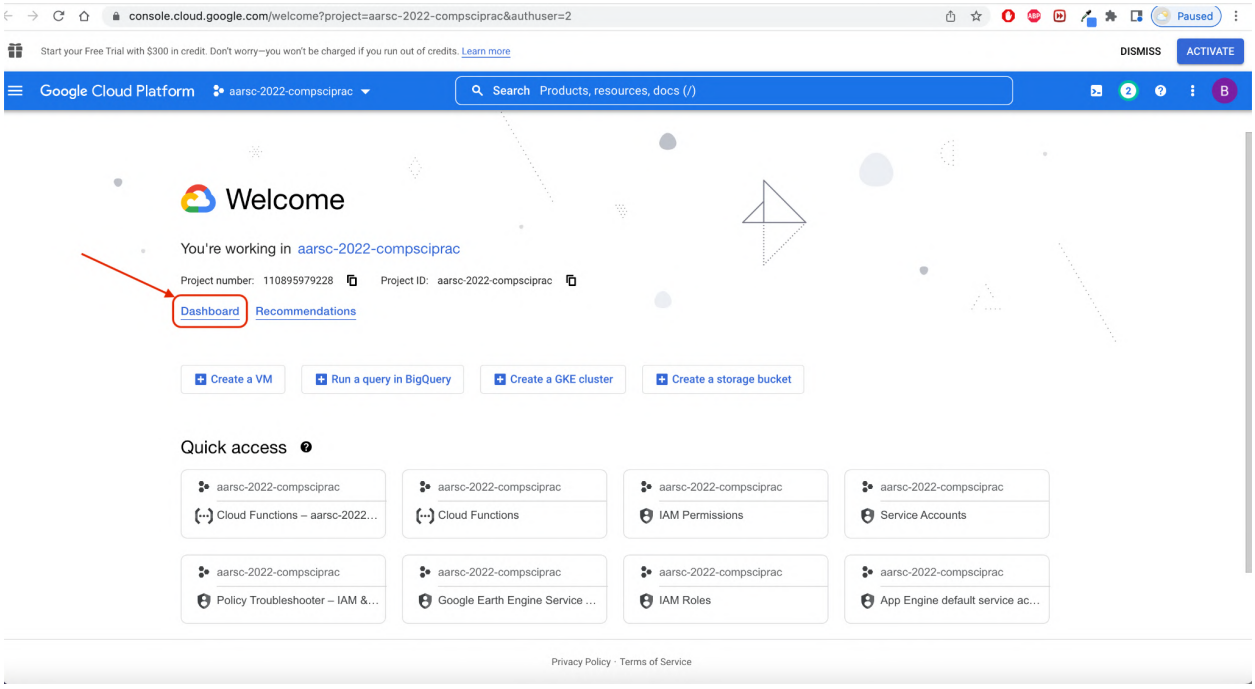
  https://console.cloud.google.com/welcome?project=aarsc-2022-compsciprac

  If you are not logged into an account with access permission, you might see the below screen that has a "select project" drop down at the left of the search bar":

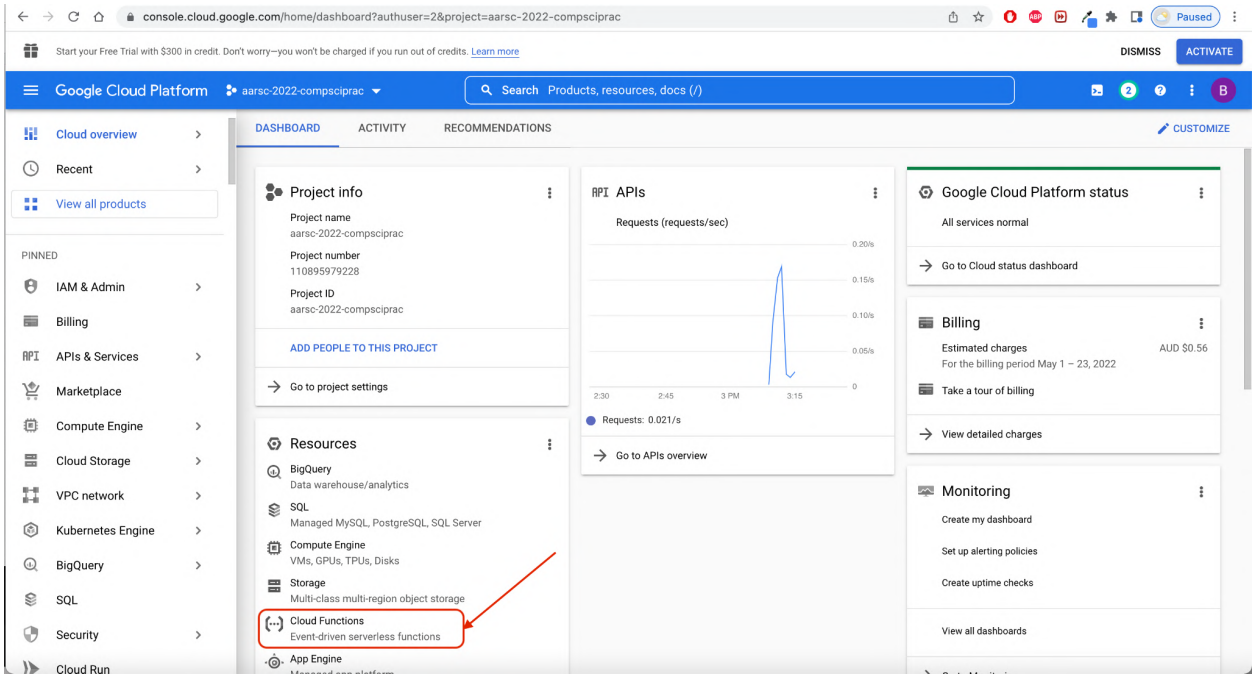


`Tip:` If you see something similar to the above page, please log into an account with access to the project.

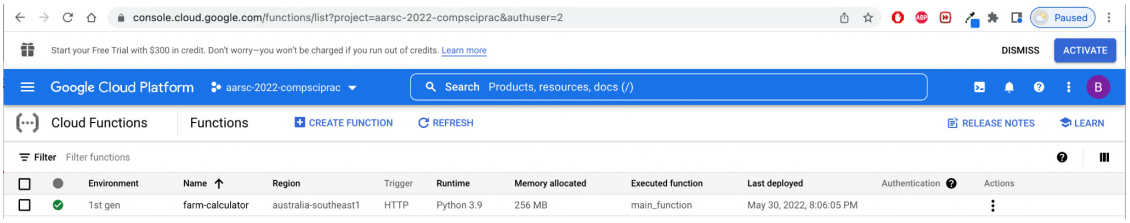
- When logged into an account with access to the project, you will see the project name loaded within the different services; please navigate to the dashboard and click on the link:



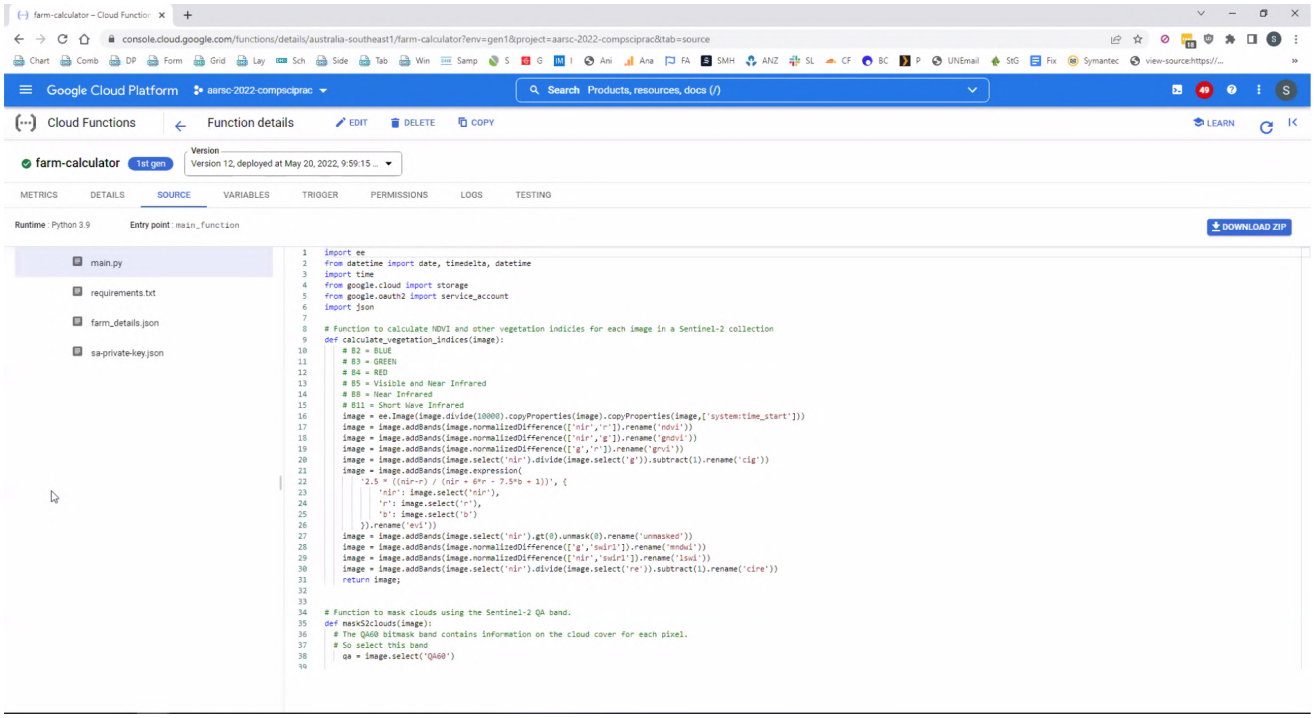
- While on the dashboard, head over to the “Cloud Functions” service under the “Resources” section, and click on it:



- Next; head over to the existing functions and select the "farm-calculator" function:



- Once clicked and after the new page loads up, navigate to and click on the "Source" tab, it is two tabs to the right of the "Metrics" tab; On the left side of the screen, we can see 4 files, namely: "main.py", "requirements.txt", "farm_details.json" , and "sa-private-key.json.



- **Main.py** – This file contains the python code separated into various functions.

  The below code highlighted in red from line 1 to line 6, are the package and library imports utilised by the main function python script of the project;

  While, the below *code highlighted in purple from line 9 to line 31*, is the function used to calculate the vegetation indices of interest and assign a corresponding color to the bands of data.

  These values and assignments can be changed to accommodate other vegetation indices, and re-assignment of corresponding colors for the bands of data to be collected.

```
import ee
from datetime import date, timedelta, datetime
import time
from google.cloud import storage
from google.oauth2 import service_account
import json

# Function to calculate NDVI and other vegetation indicies for each image in a Sentinel-2 collection
def calculate_vegetation_indices(image):
    # B2 = BLUE
    # B3 = GREEN
    # B4 = RED
    # B5 = Visible and Near Infrared
    # B8 = Near Infrared
    # B11 = Short Wave Infrared
    image = ee.Image(image.divide(10000).copyProperties(image).copyProperties(image,['system:time_start']))
    image = image.addBands(image.normalizedDifference(['nir','r']).rename('ndvi'))
    image = image.addBands(image.normalizedDifference(['nir','g']).rename('gndvi'))
    image = image.addBands(image.normalizedDifference(['g','r']).rename('grvi'))
    image = image.addBands(image.select('nir').divide(image.select('g')).subtract(1).rename('cig'))
    image = image.addBands(image.expression(
        '2.5 * ((nir-r) / (nir + 6*r - 7.5*b + 1))', {
            'nir': image.select('nir'),
            'r': image.select('r'),
            'b': image.select('b')
        }).rename('evi'))
    image = image.addBands(image.select('nir').gt(0).unmask(0).rename('unmasked'))
    image = image.addBands(image.normalizedDifference(['g','swir1']).rename('mndwi'))
    image = image.addBands(image.normalizedDifference(['nir','swir1']).rename('lswi'))
    image = image.addBands(image.select('nir').divide(image.select('re')).subtract(1).rename('cire'))
    return image;


```

- Below the calculation of vegetation indices, is the below code from line 35 to line 61the ‘maskS2clouds’ and ‘mask_clouds’ functions contained in the orange rectangle are used to mask clouds using the Sentinel-2 QA band and the Sentinel-2 collections; while, the code from line 64 to line 70 contained in the blue rectangle, is a function called “get_sentinel2_image_collection()” that takes in a start_date, end_date and an image collection, which it uses to select and rename relevant bands of interest to easily readable values. The mask values and collections can be tweaked, and the relevant bands selected can be changed and renamed as per the interests of the user.

```
# Function to mask clouds using the Sentinel-2 QA band.
def maskS2clouds(image):
  # The QA60 bitmask band contains information on the cloud cover for each pixel.
  # So select this band
    qa = image.select('QA60')

  # Bits 10 and 11 are clouds and cirrus, respectively.
  # Use zero fill left shift operators
    cloudBitMask = 1 << 10
    cirrusBitMask = 1 << 11

  # Both flags should be set to zero, indicating clear conditions.
    mask = qa.bitwiseAnd(cloudBitMask).eq(0) and qa.bitwiseAnd(cirrusBitMask).eq(0)

  # Return the masked and scaled data, without the QA bands.
  # The value returned is either null if clouds are obscuring the
  # pixel or the actual value if not
    # return image.updateMask(mask).divide(10000).select("B.*").copyProperties(image, ["system:time_start"])
    return image.updateMask(mask)


def mask_clouds(img):
    # masks clouds in Sentinel-2 collections
    img=ee.Image(img)
    clouds = ee.Image(img.get('cloud_mask')).select('probability')
    isNotCloud = clouds.lt(40)
    return img.updateMask(isNotCloud)


# select the relevant bands and rename to more human-readable heading values
def get_sentinel2_image_collection(start_date,end_date, coll='COPERNICUS/S2_HARMONIZED'):
    sentinel2_image_collection = ee.ImageCollection(coll) \
        .filterDate(start_date,end_date) \
        .select(['B2','B3','B4','B5','B8','B6','B7','B8A','B11','B12','QA60'], ['b','g','r','re','nir','re74','re78','re86','swir1','swir2','QA60'])

    sentinel2_cloud_probability = ee.ImageCollection('COPERNICUS/S2_CLOUD_PROBABILITY')\
        .filterDate(start_date,end_date)

```

- Below, the Mask cloud, collection selection-and-renaming functions, are the below functions:

  The “get_timeseries_feature_collection()” function is highlighted within the yellow rectangle in the below image and spans code from line 81 to line 97; It is used to get the time series over the feature collection and then return the image collection sample images. It takes in a ‘feature collection’, image collection, jsonfile (geojson) image properties, and scale as arguments.

  The function below the “get_timeseries_feature_collection()” function is the “add_date_to_image()” function highlighted in the green rectangle in the image below. It takes in an image as an argument and allows the user to choose a date format and add the corresponding dates to the images.

  Both of these functions allow changes to be made to the arguments passed in, the scale of the images, and the format of the date that is appended to the images

```
71
72      sentinel2_image_collection = ee.Join.saveFirst('cloud_mask').apply(**{
73        'primary': sentinel2_image_collection,
74        'secondary': sentinel2_cloud_probability,
75        'condition': ee.Filter.equals(leftField='system:index', rightField='system:index')
76      })
77      return ee.ImageCollection(sentinel2_image_collection).map(maskS2clouds).map(calculate_vegetation_indices)
78
79
80  #get time series over a feature collection, general, doesn't filter ic
81  def get_timeseries_feature_collection(feature_collection,image_collection,json_file,image_props=[],scale=10):
82      def get_time_series_single_feature(feature):
83          geom = feature.geometry();
84          def sample_image(image):
85              return feature.set(image.reduceRegion(**{'reducer':ee.Reducer.mean(), 'geometry':geom, 'scale':scale})) \
86                  .setGeometry(geom.centroid(10)) \
87                  .set(image.toDictionary(image_props)) \
88                  .set('area_gee',geom.area(1).divide(10000)) \
89                  .set('geojson_filename',json_file)
90          samples = image_collection.map(sample_image)
91          return samples
92
93      #get time series for a single feature
94      time_series = feature_collection.map(get_time_series_single_feature)
95
96      #will be a collection of collections, so need to flatten to a collection
97      return time_series.flatten()
98
99
100 def add_date_to_image(image):
101     tz = 'Australia/Sydney'
102     date = ee.Date(image.get('system:time_start'))
103     au_formatted_date = date.format(**{'format':'yyyy-MM-dd','timeZone':tz})
104     return image.set({
105         'date':au_formatted_date,
106       });
107
```

The “main_function” function handles the initialisation of the service account and private key, it also contains the setting for the timeframe or time period which CropWatch-RDSIF will collect data within. It includes the loading of the geojson file to obtain the farm details from, and finally; a function to to add a description, set an output destination (databucket), and define the file format of the export (in this case ‘.csv’).

Line 109 to line 116, is where the service account and private key for the project is initialised. - This can be edited to initialise another project and another private key if needed.

```
109 def main_function(request):
110   #initialize earth engine using json key for service account
111     service_account = 'google-earth-engine-service-ac@aarsc-2022-compsciprac.iam.gserviceaccount.com'
112     privateKey = 'sa-private-key.json'
113
114     # authenticate using service account
115     credentials = ee.ServiceAccountCredentials(service_account, privateKey)
116     ee.Initialize(credentials)
```

Line 119 -120 includes a variable called “start_date”, which sets the start of the timeframe for the Google Earth images data. The variable “end_date” determines the most current date you want the function to retrieve data up to:

```
116     # set the start and end dates for the code
117     # this should be moved to arguments in the future
118     # start with a 2 year timeframe (730 days)
119     start_date = (date.today() - timedelta(days=730)).strftime("%Y-%m-%d")
120     end_date = date.today().strftime("%Y-%m-%d")
121
```

“start_date” is currently set in this screenshot to today minus 730 days (2 years) but can be adjusted to any other value. The currently deployed, default adjusted value was set to 1095 days (3 years) as can be seen in the screenshot below:

```
128     # set the start and end dates for the code
129     # this should be moved to arguments in the future
130     # start with a 3 year timeframe (1095 days)
131     start_date = (date.today() - timedelta(days=1095)).strftime("%Y-%m-%d")
132     end_date = date.today().strftime("%Y-%m-%d")
```

Hence; 3 years of data from the current date can be obtained and exported by CropWatch-RDSIF’s farm-calculator function.

The code on *line 118 to line 126*, provides the option to get an optional POST argument for the json file name:

```
118     # get optional POST argument for the json filename
119     request_json = request.get_json(silent=True)
120     request_args = request.args
121     if request_json and 'json_filename' in request_json:
122         json_filename = request_json['json_filename']
123     elif request_args and 'json_filename' in request_args:
124         json_filename = request_args['json_filename']
125     else:
126         json_filename = 'farm_details.json'
```

The code on line *134 to line 138,* allows the loading of the geojson file that is passed into the previous argument.

```
134     # load a geojson file to get farm details
135     # json_filename = 'farm_details.json'
136     json_file = open(json_filename)
137     feature_collection_json = json.load(json_file)
138     json_file.close()
```

The code on line 140 to line 151, handles the creation of the feature collection based on the geojson data supplied from the geojson file. The farm boundaries are then extracted from this feature collection and used in combination with Sentinel2 satellite images within the earlier supplied date range (start_date , end_date) to obtain the image statistics at each time for each element in the feature collection.

```
140     # create the feature collection based on the geojson supplied
141     feature_collection = ee.FeatureCollection(feature_collection_json)
142
143     # extract the farm boundaries from the feature collection
144     geom = feature_collection.geometry().bounds()
145
146     # create an image collection from the sentinel2 satellite images using the date ranges and geo boundaries
147     # - collection of S2 images covering the feature collection of interest
148     sentinel2_image_collection = get_sentinel2_image_collection(start_date,end_date).filterBounds(geom).map(add_date_to_image)
149
150     #gets image statistics at each time over each field defined in feature collection
151     time_series = get_timeseries_feature_collection(feature_collection,sentinel2_image_collection,json_filename,image_props=['date','ndvi','gndvi','grvi','cig','mndwi','lswi','cire'],scale=10)
```

The code on line 156 sets the description field for the data export. The system utilises this field to set the name of the exported CSV file. It is currently set to farm_data_export_[current date/time]; however, this can be changed to any other value.

The code from line 154 to line 163 is utilised to add a description field for the data being exported, set an output destination (databucket), and define the file format of the export (in this case '.csv') as can be observed below:

```
153     # Function to export statistical values per ID with the information of day, month and year
154     task = ee.batch.Export.table.toCloudStorage(
155         collection=time_series,
156         description='farm_data_export_'+str(datetime.now().year) + "-" + str(datetime.now().month) + "-" + str(datetime.now().day) + " " + str(datetime.now().hour) + ":" + str(datetime.now().minute),
157         outputBucket='aarsc-2022-compsciprac-csv-databucket',
158         fileFormat= 'csv'
159         #selectors=['date','id','ndvi','gndvi','grvi','cig','mndwi','lswi','cire']
160     )
161     task.start()
162
163     return "completed"
164
```

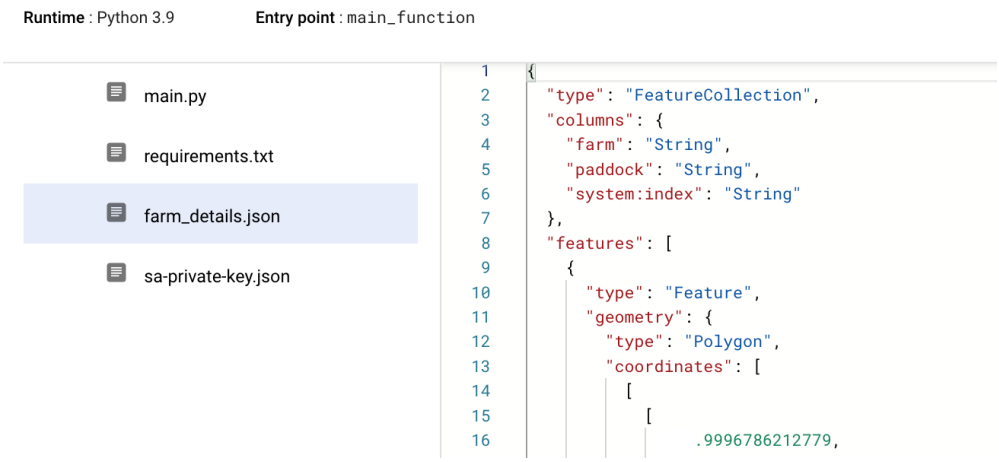
- Once the file is exported, the filename is set and exported to the databucket as: 'farm_data_export_[date and time of the export] in a '.csv format.

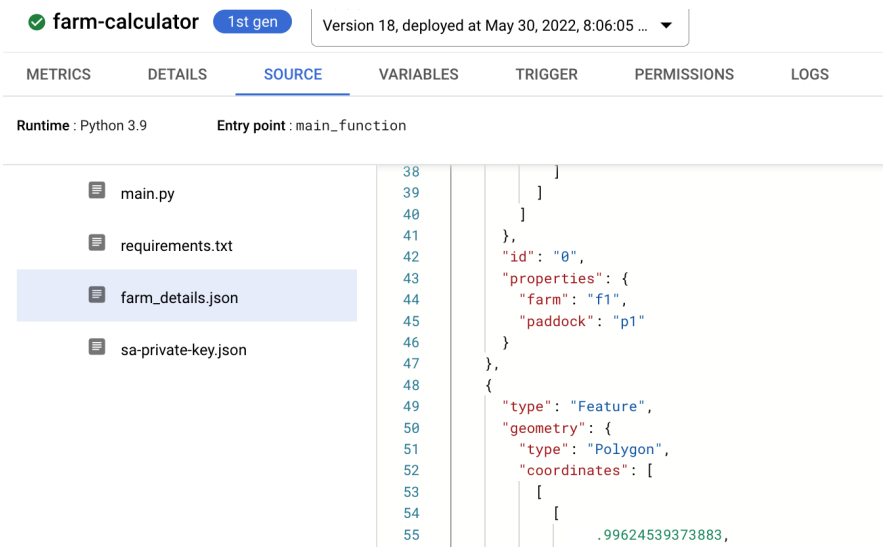
The other documents within the farm-calculator are as follows:

- **Requirements.txt** – This file sets out the relevant dependencies for the Python code. Currently, the code relies on the Google Earth Engine API and Google Cloud Storage

- **Farm_details.json** – This file sets out the boundaries of the farms to be examined with the code. This file contains GeoJSON data saved in JSON format.

  Coordinates redacted example:





To examine a different set of farms, this file can be either replaced with another valid FeatureCollection in JSON format or adjustments to the python code on line 123, within the main function can be made.

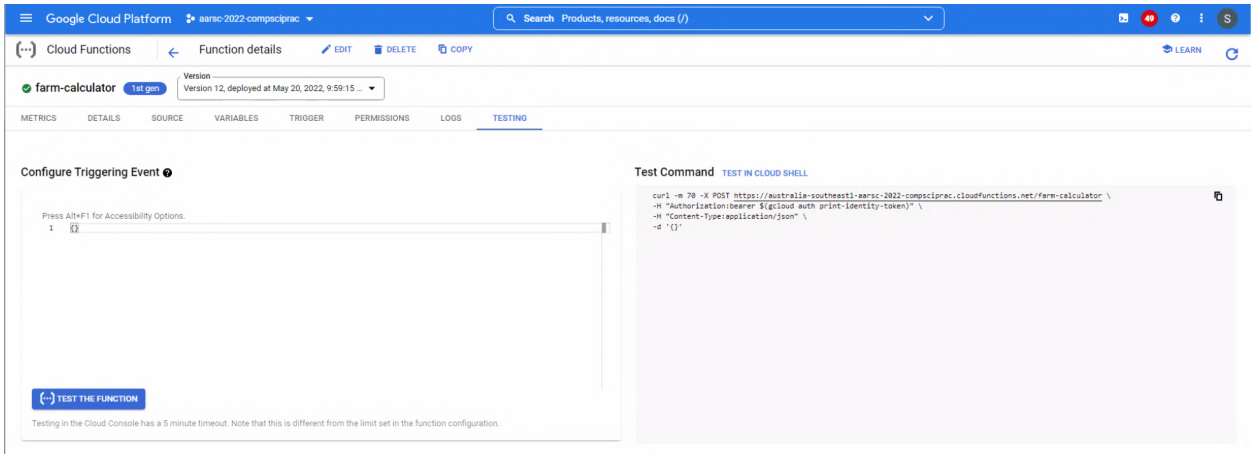
- **Sa-private-key** – This file provides the private key for the service account used to run the Python code. If a different service account is used, this file will need to be updated.

  `Tip:` This "Sa-private-key" file, is a confidential file of which its contents cannot be shown to non-members of the team and should not be uploaded to the open source github repository.

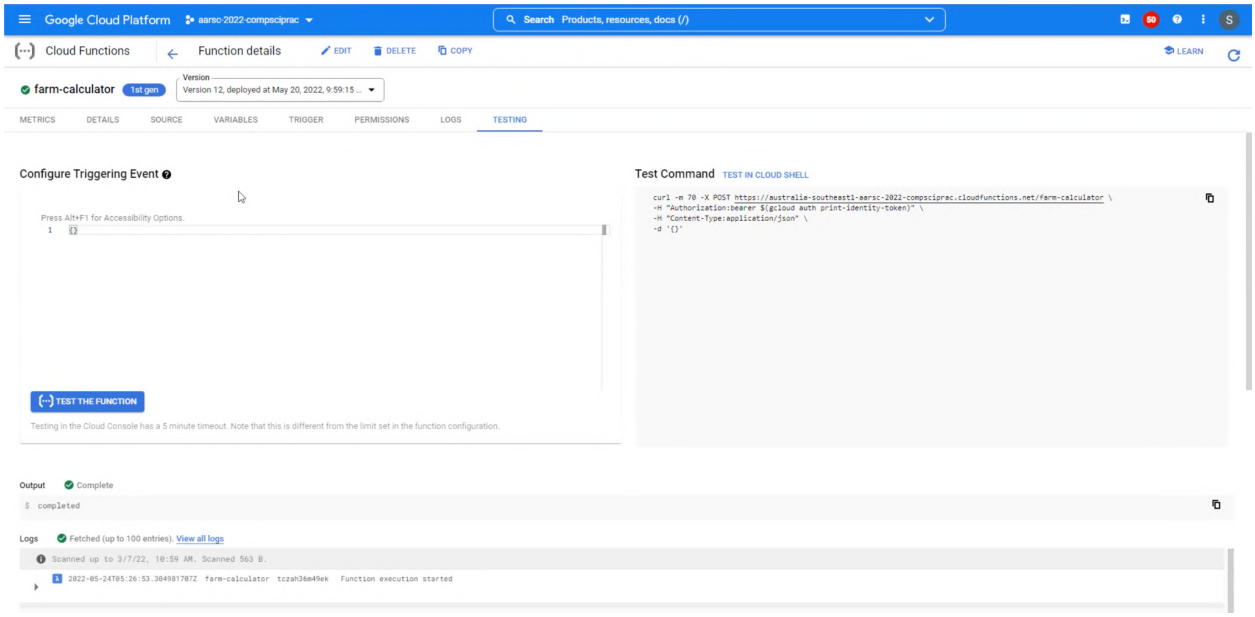
There are a number of ways to run the cloud function:

1. Use the HTTP trigger URL (https://australia-southeast1-aarsc-2022-compsciprac.cloudfunctions.net/farm-calculator). Permissions are required to run the function this way; otherwise, access will be denied.

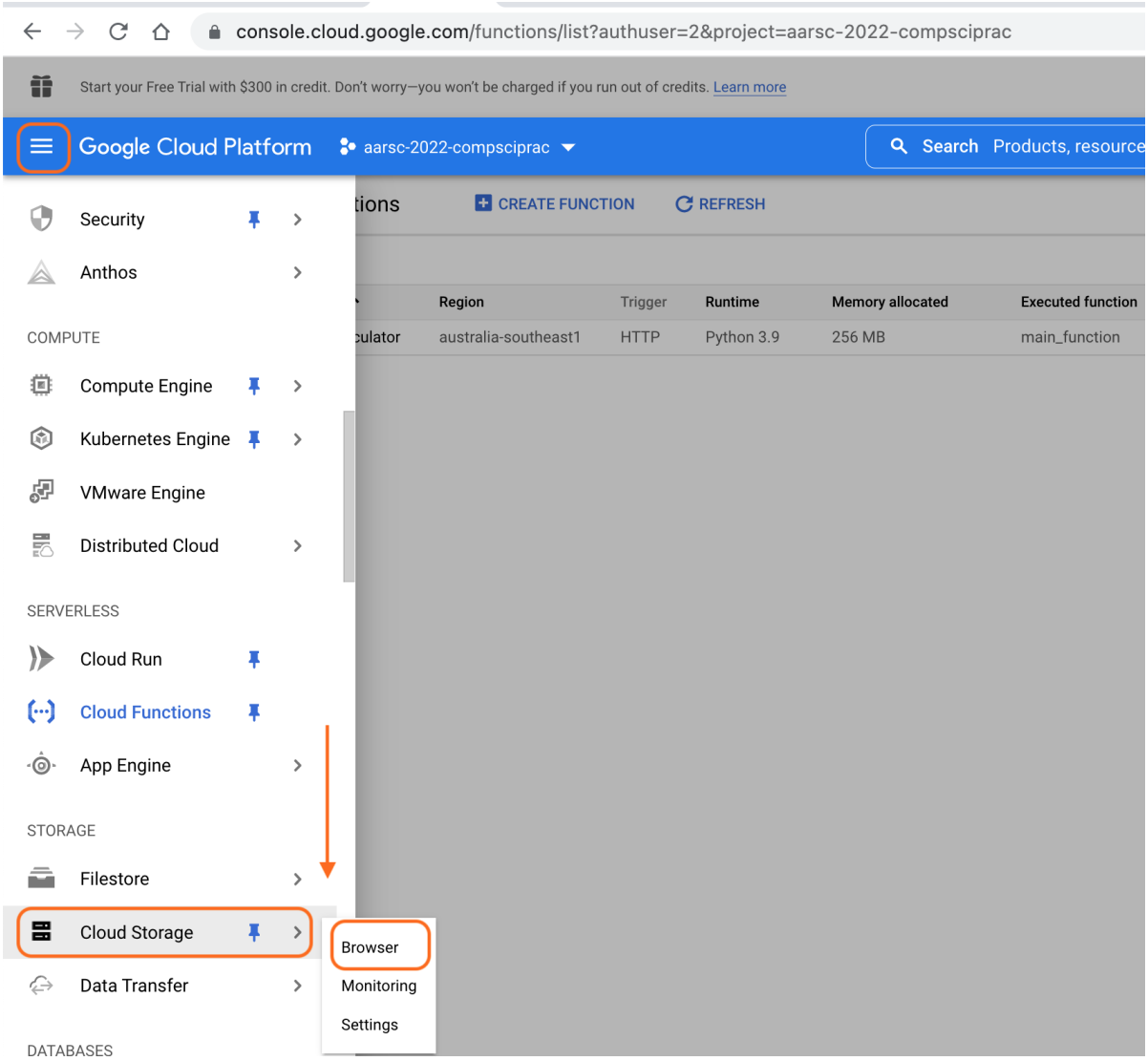
2. Open the function, select the 'Testing' tab and then click the 'Test the Function' button .



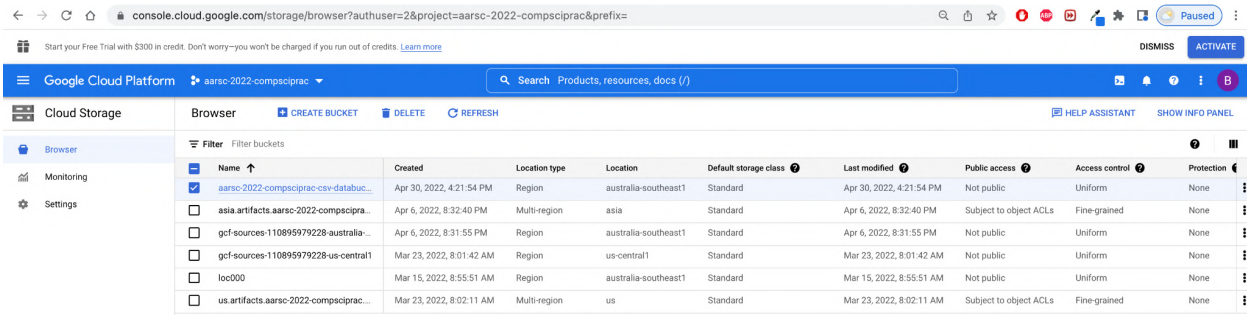
- Below is what shows up when the task is completed



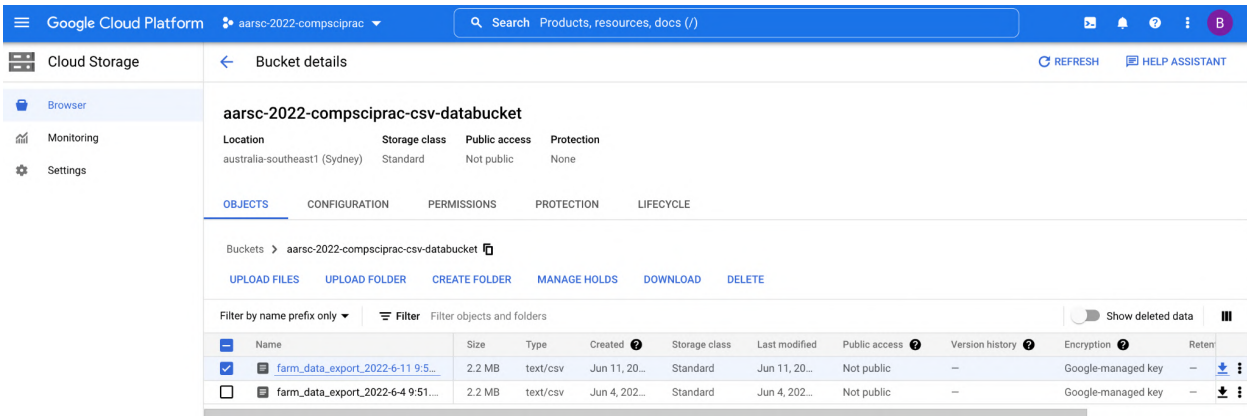
- Next, navigate to the cloud storage (databucket) by clicking the menu icon at the top left of the page, next to “Google Cloud Platform” header, then scroll down to the “Storage” section and click on “Cloud Storage”, then “browse”:



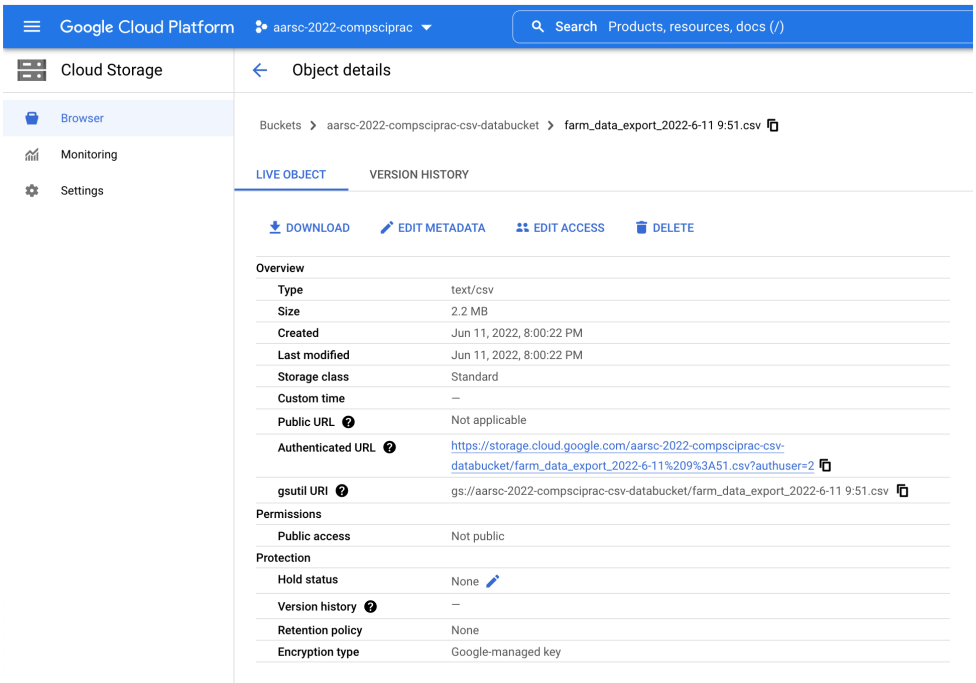
- This will load the cloud storage page which will have the databucket that is being used within the project at the top of the list (only selected below for reference purposes), click on it:



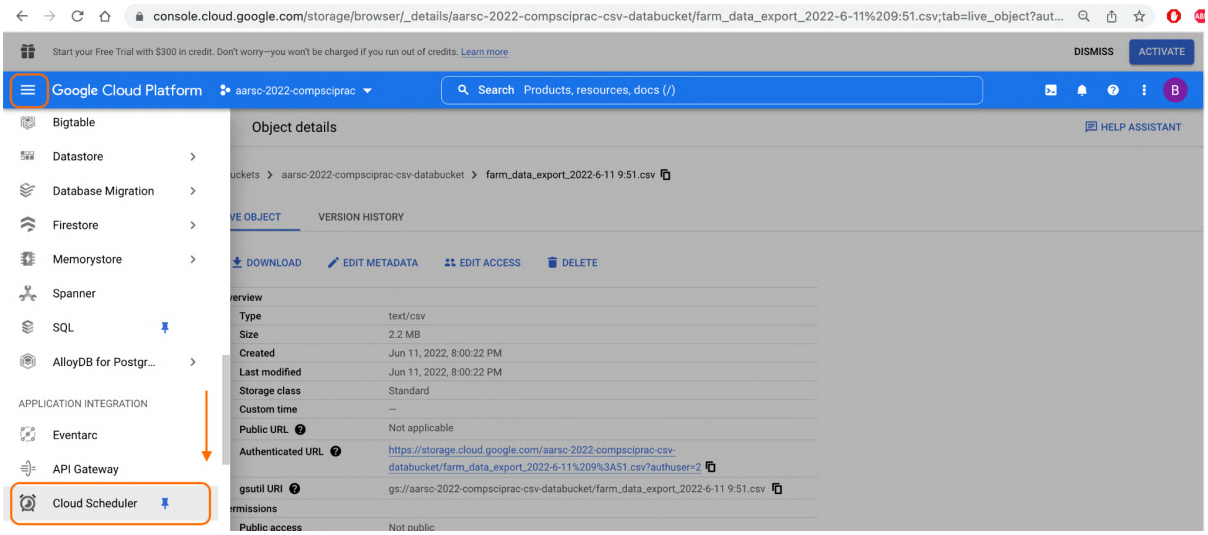
- The page that loads has the most current exported data (only selected below for reference purposes) (based on service usage, the time it takes to appear in the cloud storage might vary, based on if you are exporting at peak hours and or if your internet connection is not as fast):



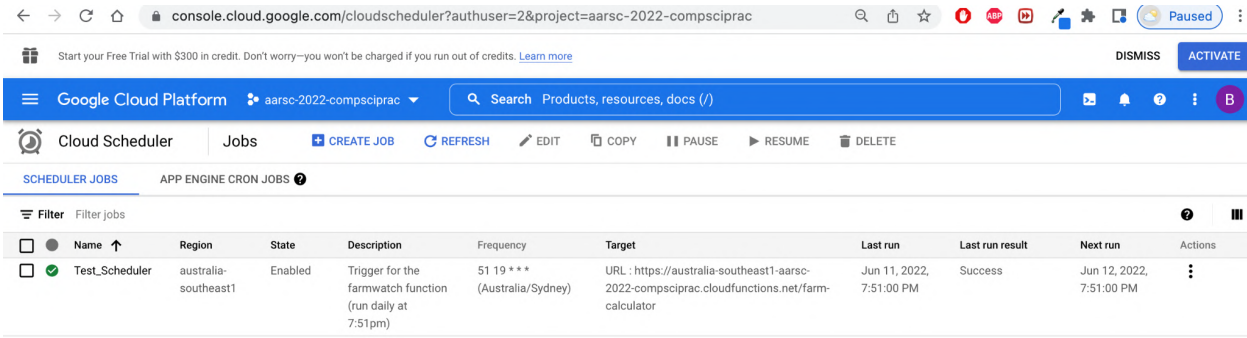
Clicking on the file, shows further information about the file as can be seen below, and provides the option to edit access, change metadata, delete the file or to download the file:



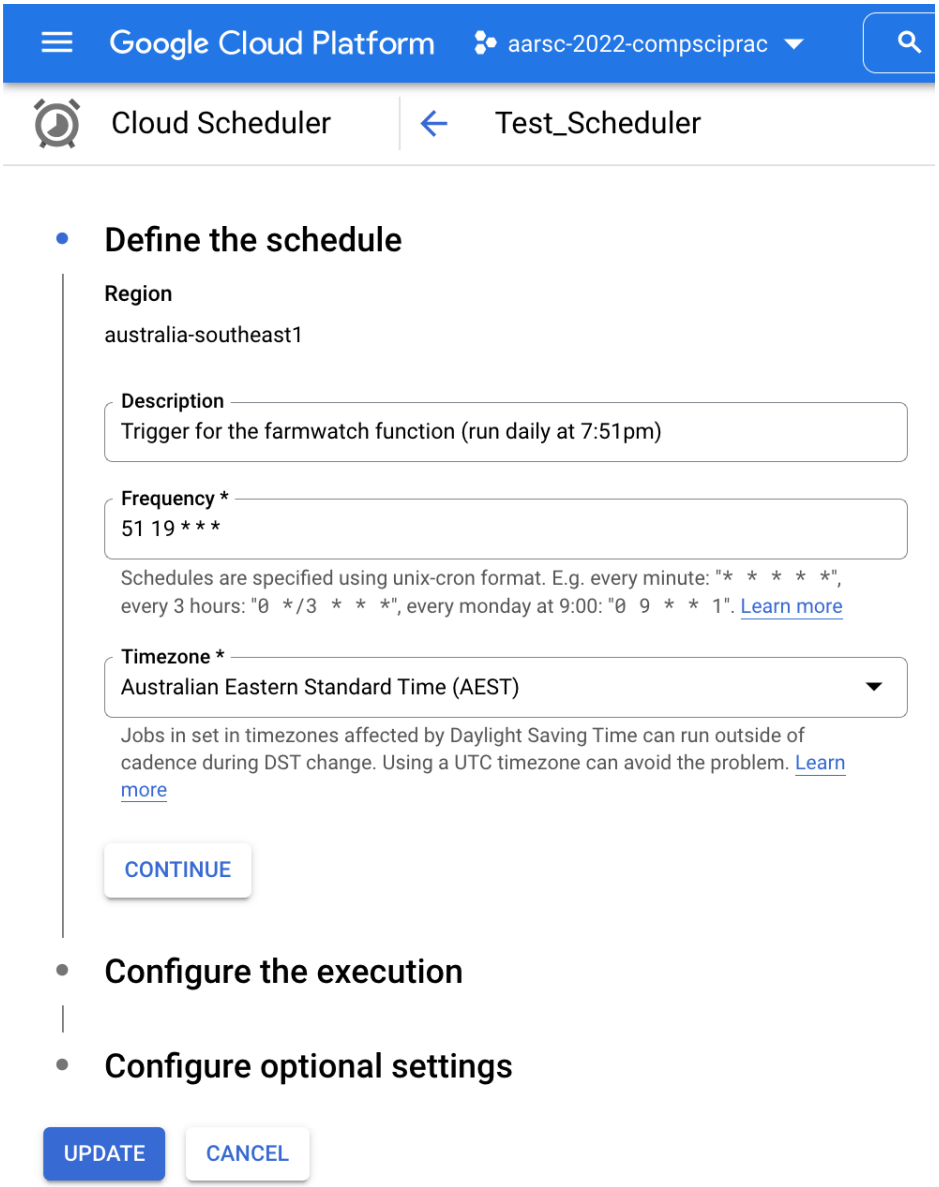
3. The files are processed using Google Cloud Scheduler jobs. A scheduled task has been set up to run the function daily; however, the scheduler will also allow a manual trigger of the task by clicking the 'Run Now' button.
    - Click on the menu icon at the top left of the page, next to "Google Cloud Platform" header, then scroll down to the "Application Integration" section and click on "Cloud Scheduler":



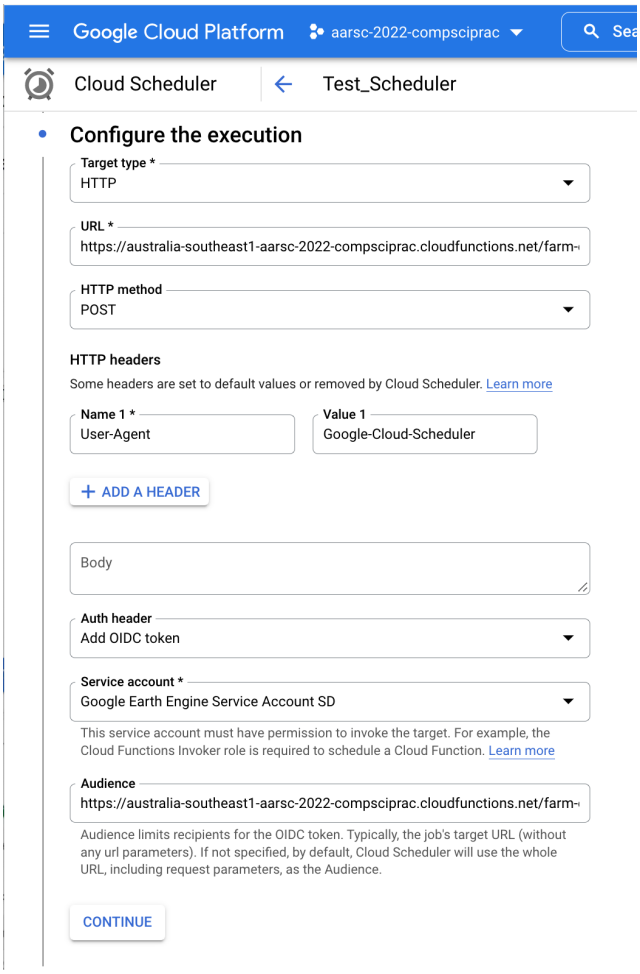
    - Next, the page loads up and shows the aforementioned existing schedule that has already been set up:



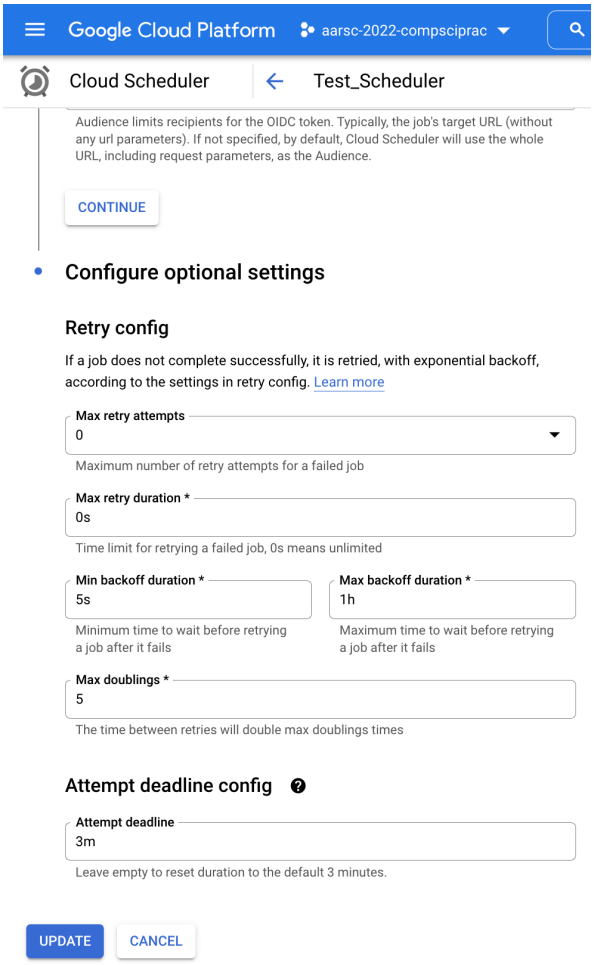
    - Changes can be made to this schedule by clicking on it and making adjustments to the parameters below, please observe that the cloud scheduler in this "test_schedule" is currently set to trigger and run the CropWatch farm-calculator function every day at 7:51 pm, AEST (Australian Eastern Standard Time):



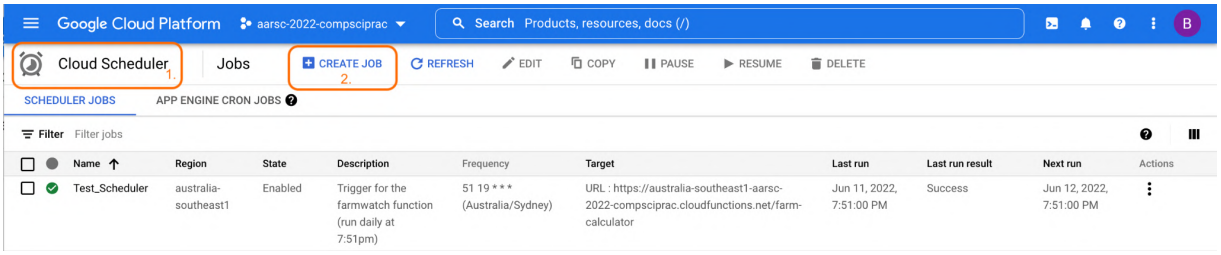
- The current execution configuration of the cloud scheduler is displayed below, and can be adjusted as pleased:



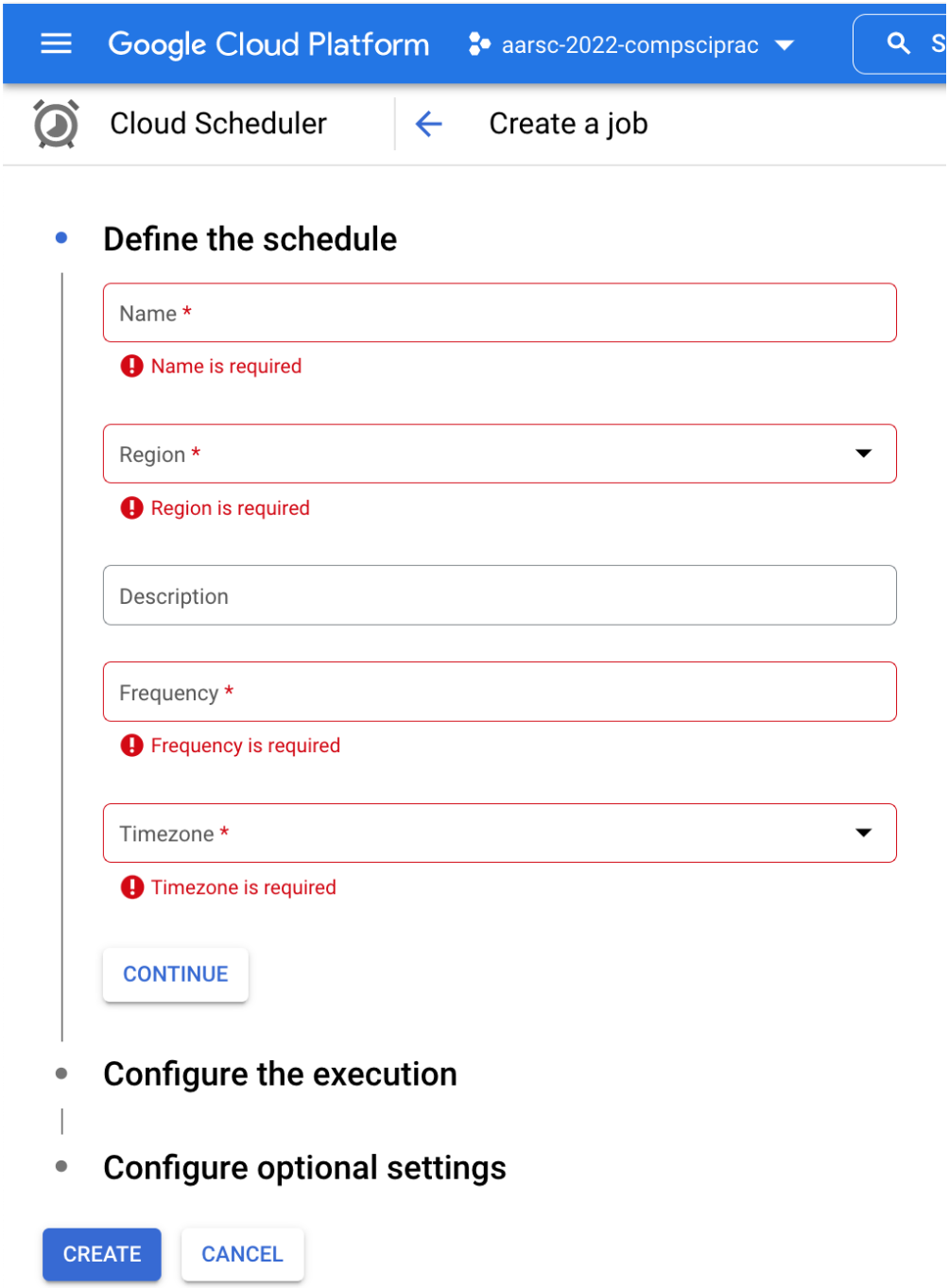
- The optional settings that handle the retrying of the scheduled function in the event of a job not completing successfully are shown below; and can be adjusted as pleased:



- To create a new job; click on the “Cloud Scheduler” tab at the top left of the page, then click on the “Create Job” tab located at the top, near the middle of the page:



- Next you will be provided an empty form to fill in the parameters with required information to create the scheduled job (**Please note that:** an example of the required information needed, has already been shown in the above steps within the above existing scheduled job named “Test_Scheduler”):



This concludes the options and processes directly related with the cloud-function.

CropWatch-RDSIF: Exporting farm data to the Database (BigQuery)

After data gets exported from the CropWatch-RDSIF farm-calculator's Main.py script's main function into the "Aarsc-2022-compsciprac-csv-databucket", the project's Google Cloud Storage Bucket; the below processes commence:

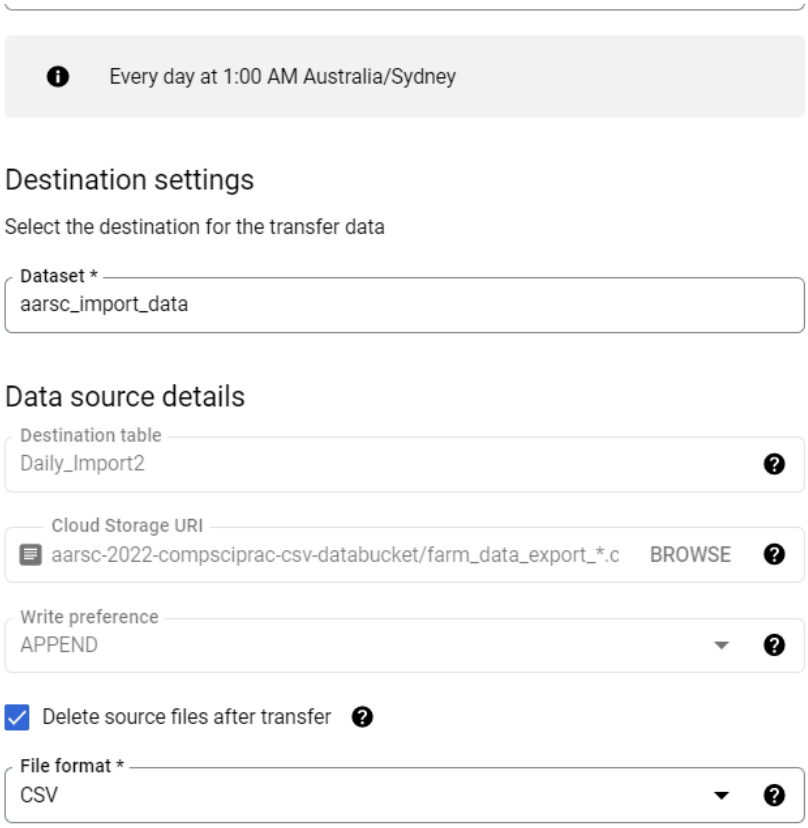
Daily Data Transfer from the Databucket

A daily Google Data Transfer called "Daily Load 2" runs at 1:00 AM AEST and transfers all files that meet the following criteria
gs://aarsc-2022-compsciprac-csv-databucket/farm_data_export_*.csv to table

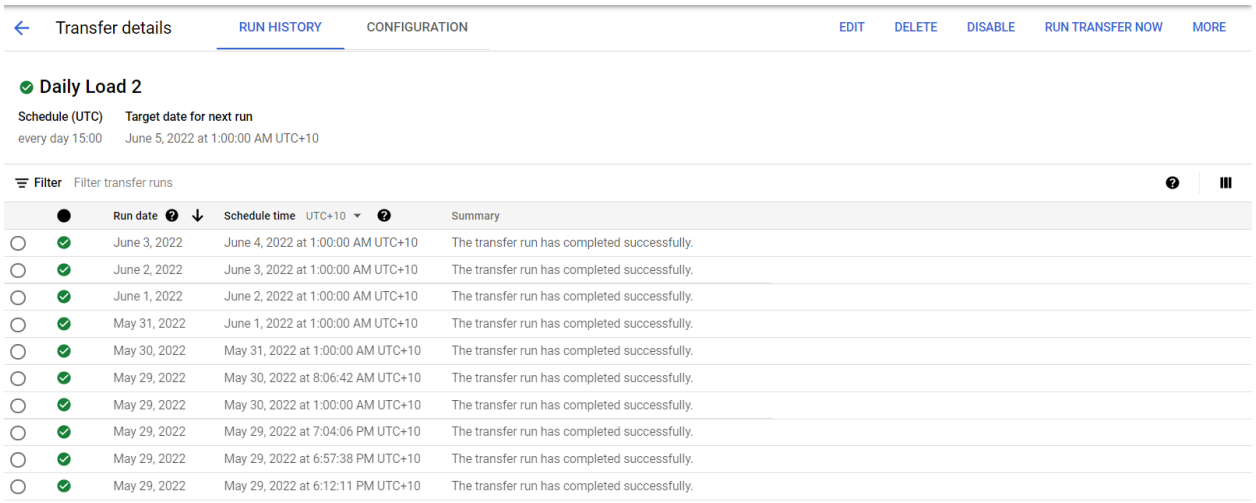
`aarsc-2022-compsciprac.aarsc_import_data.Daily_Import2`

All files that meet the file name criteria are appended to the Daily_Import2 table. At this point there are no checks for duplicates carried out on the files.

Details of this transfer can be amended by clicking on the transfer, then Configuration / Edit. In this section you can select for the files not to be deleted after processing. This can be done by clicking the data transfer, clicking on Configuration/Edit and then clicking off the Delete Source Files on Transfer button. Note that if you wish to reprocess the files you will need to rename them prior to reprocessing. The timing of the transfer as well as some of the file configuration details can be altered in this section.



The data transfer can be run on demand by selecting the data transfer and then selecting Run Transfer now at the top right of the screen.

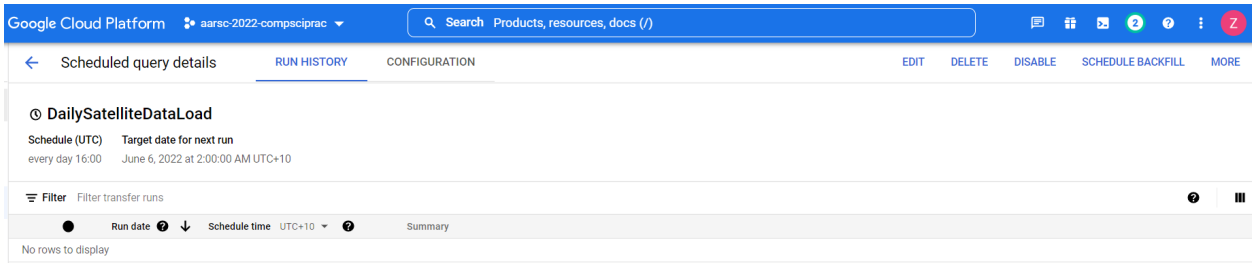


Scheduled Daily SQL

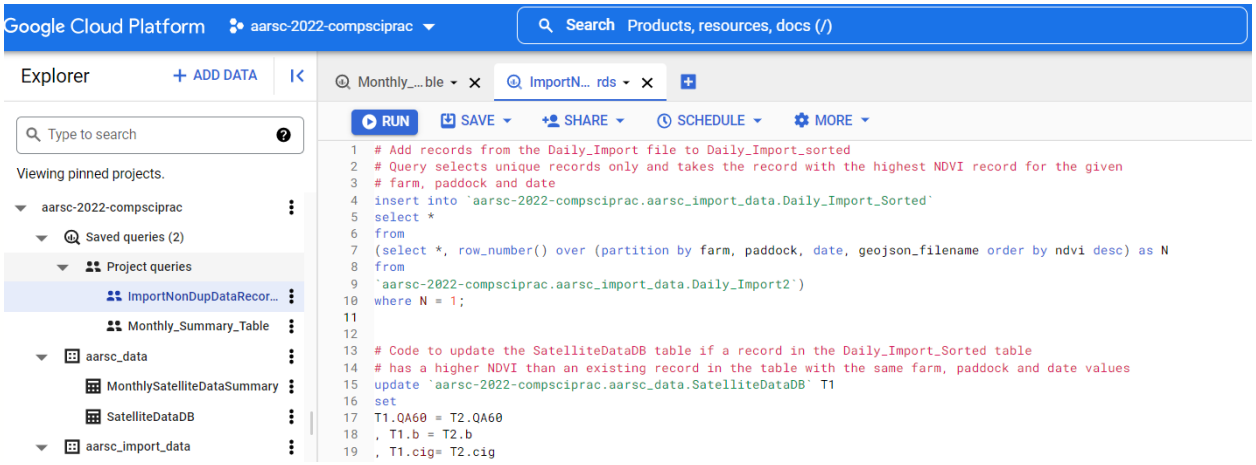
At 2:00 AM each day a scheduled SQL query runs, DailySatelliteDataLoad. It does the following:

- Selects unique records (based on farm, paddock, date adn geojson_filename) in the Daily_Import file to copy to the Daily_Import_sorted file. Where duplicates exist the record with the highest ndvi record is selected.
- For each record in the Daily_Import_sorted table a check is done to see if a record with the same farm, paddock, date and geojson_filename exists in the SatelliteDataDB table. If a record exists and the ndvi is less than the ndvi in the Daily_Import_sorted table an update is made to SatelliteDataDB to include the values for the record with the higher ndvi from Daily_Import_sorted
- For each record in the Daily_Import_sorted table if a record with the same farm, paddock, datedate and geojson_filename doesn't exist the record is added to the SatelliteDataDB table
- The content from both
  `aarsc-2022-compsciprac.aarsc_import_data.Daily_Import`
  `aarsc-2022-compsciprac.aarsc_import_data.Daily_Import_sorted`
  is deleted in preparation for the following nightly run

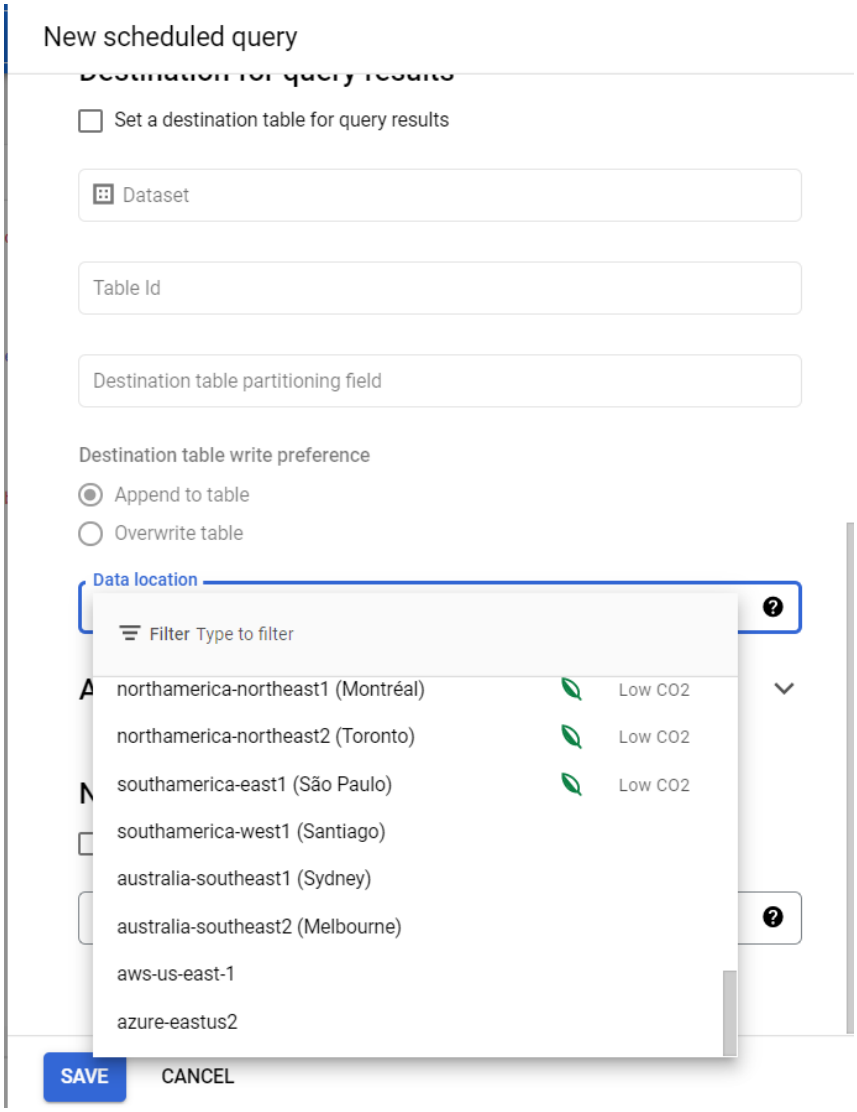
The scheduled query can be run on demand by selecting the query and then clicking on Schedule Backfill in the top right hand corner of the screen.



The underlying SQL can also be run manually by selecting the ImportNonDupDataRecords within the BigQuery database. By highlighting the section of the SQL to be run and clicking on Run at the top the SQL can be run in sections.



Please note that: changes made to the underlying SQL do not get passed through to the schedule query. To update the scheduled query you will need to create a new scheduled query by clicking on Schedule / Create new Schedule. You will also need to ensure that the data location is consistent with where the data is currently being stored, in this case australia-southeast1 (Sydney).



Scheduled Monthly SQL

At 3am on the 4th day of each month a monthly scheduled SQL is run called MonthlySummaryUpdate. This process deletes the content of the MonthlySatelliteDataSummary table and recreates it based on the data available in the SatelliteDataDB table. The summary table contains monthly averages, minimums and maximums, standard deviation and the upper and lower 95% confidence intervals for each of the indices in Satellite Data DB by farm, paddock and geojson_filename.

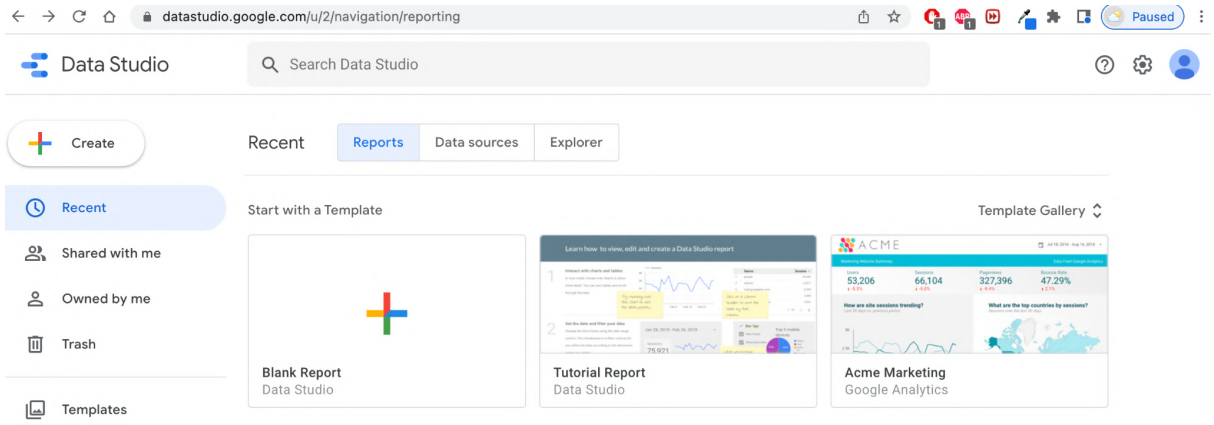
CropWatch-RDSIF: Visualizing Database information in DataStudio

Google DataStudio process

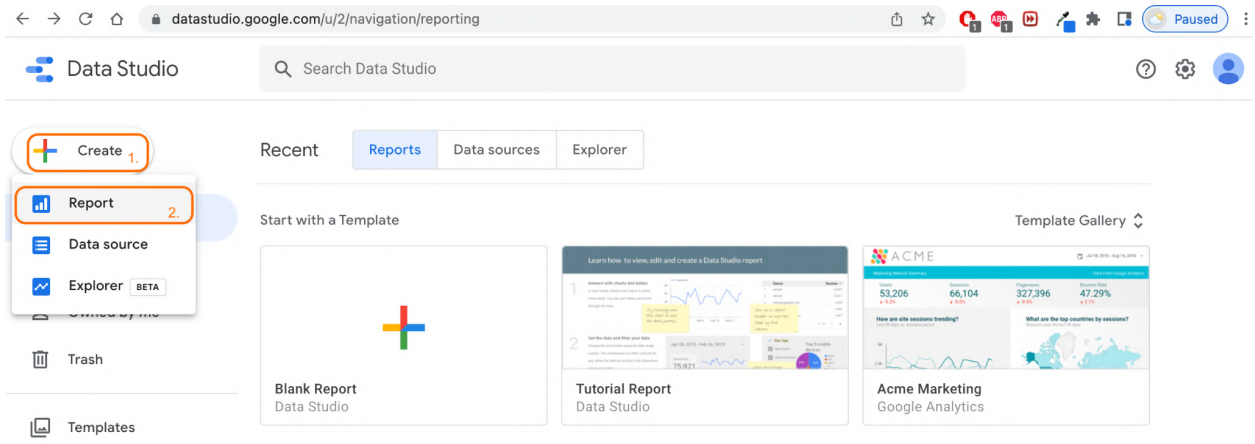
The end product of the CropWatch-RDSIF application process is to create reports that construct visual representations of the data collected and processed, to provide farmers valuable insights on the crop biomass and crop health of the crops in their paddocks/ farms over a given period of time up to 3 years front he current date.

The following steps should be followed to create these visual representations of the data collected and processed:

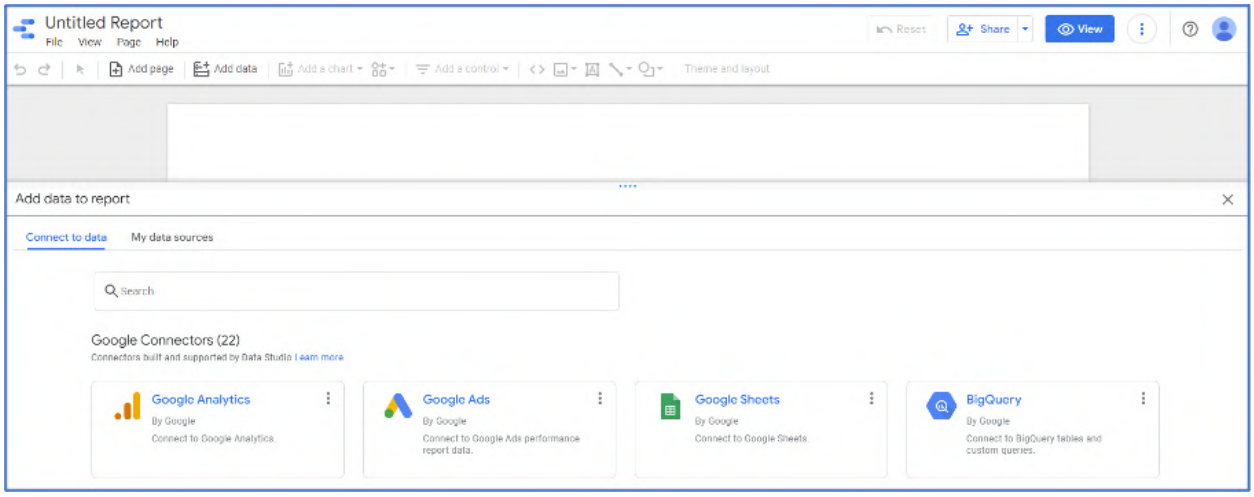
1. Go to https://datastudio.google.com/



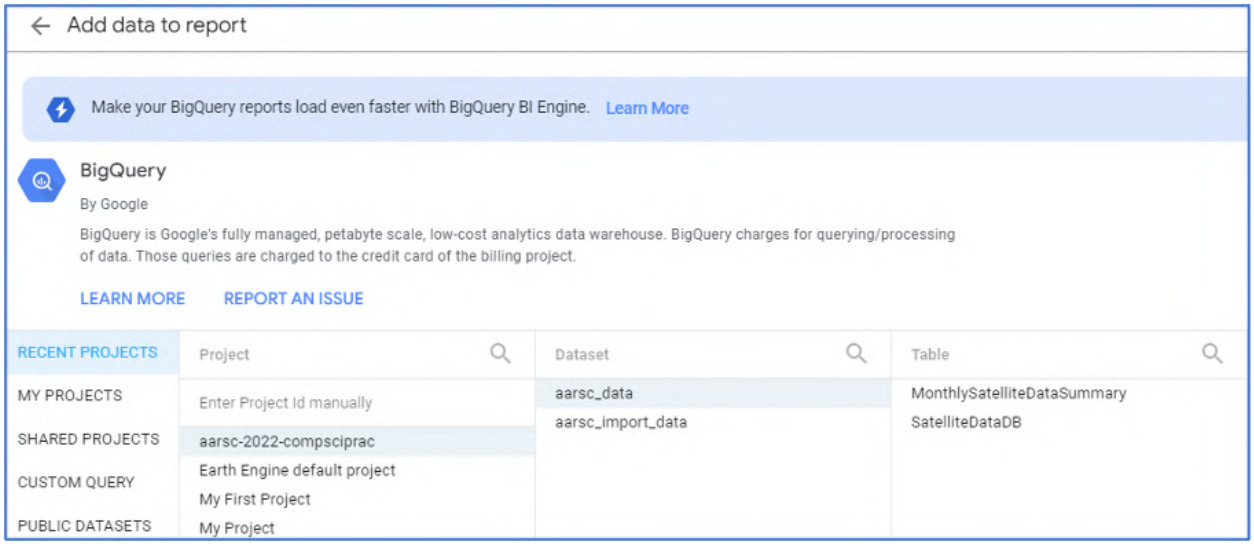
2. Click on ‘Create’ , then select ‘Report’:



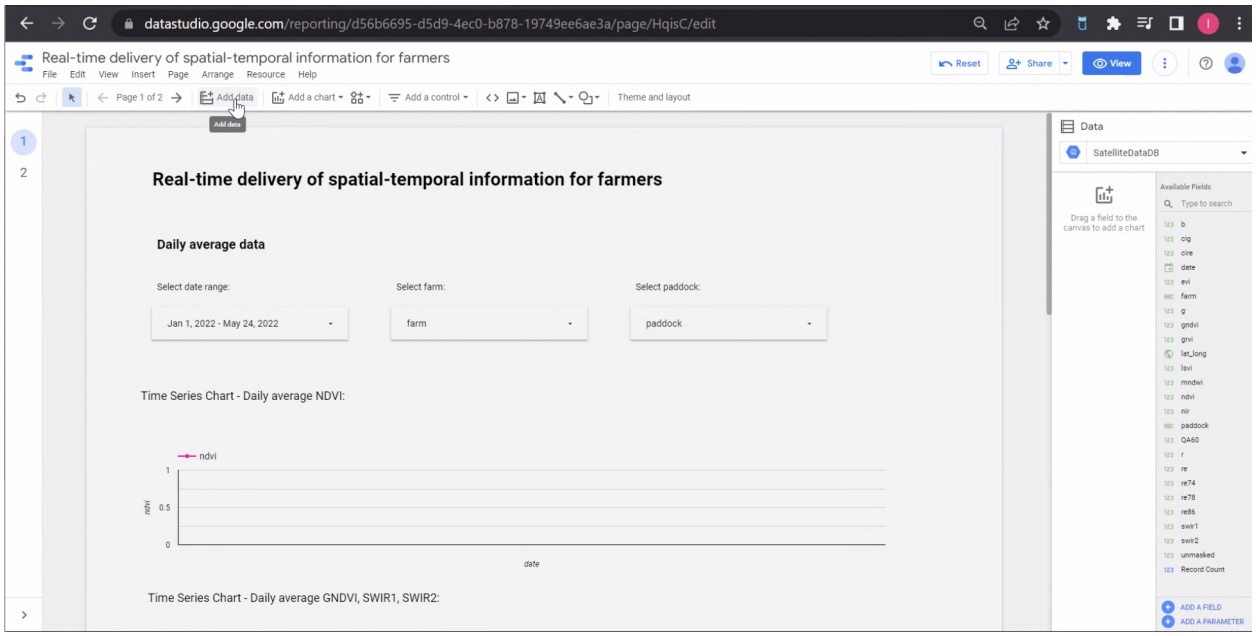
3. Add data to report, choose ‘connect to data’, then click on ‘BigQuery’.



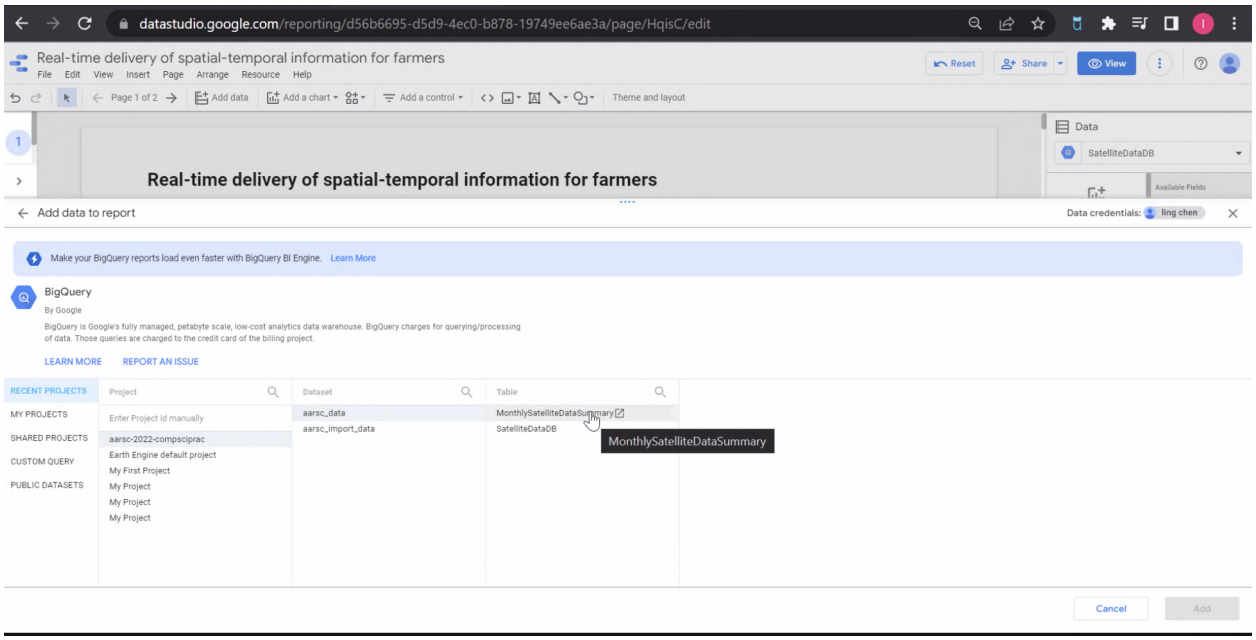
4. The select project ‘aarsc-2022-compsciprac’ ‘aarsc_data’ ‘MonthlySatelliteDataSummary’ and ‘SatelliteDataDB’



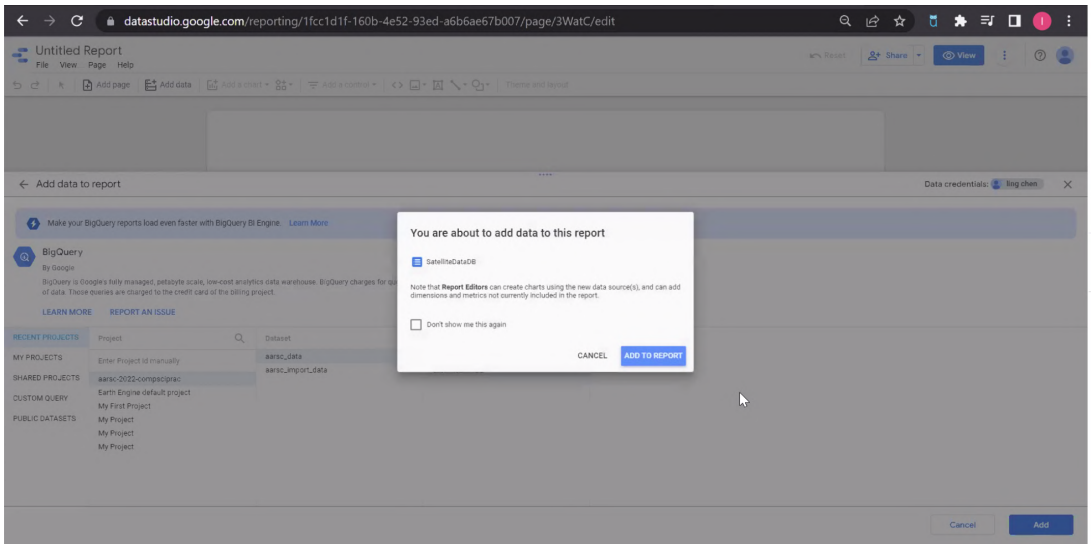
When the report is created, if the below is shown, click on the "Add data" tab at the top:



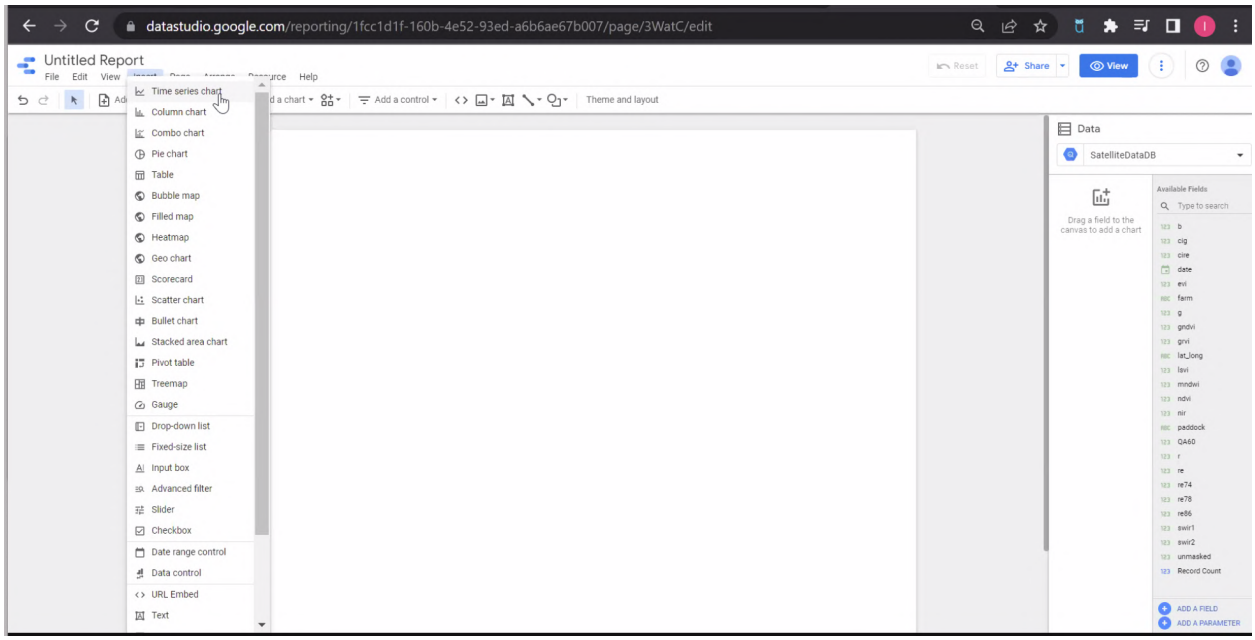
Select the data to add:



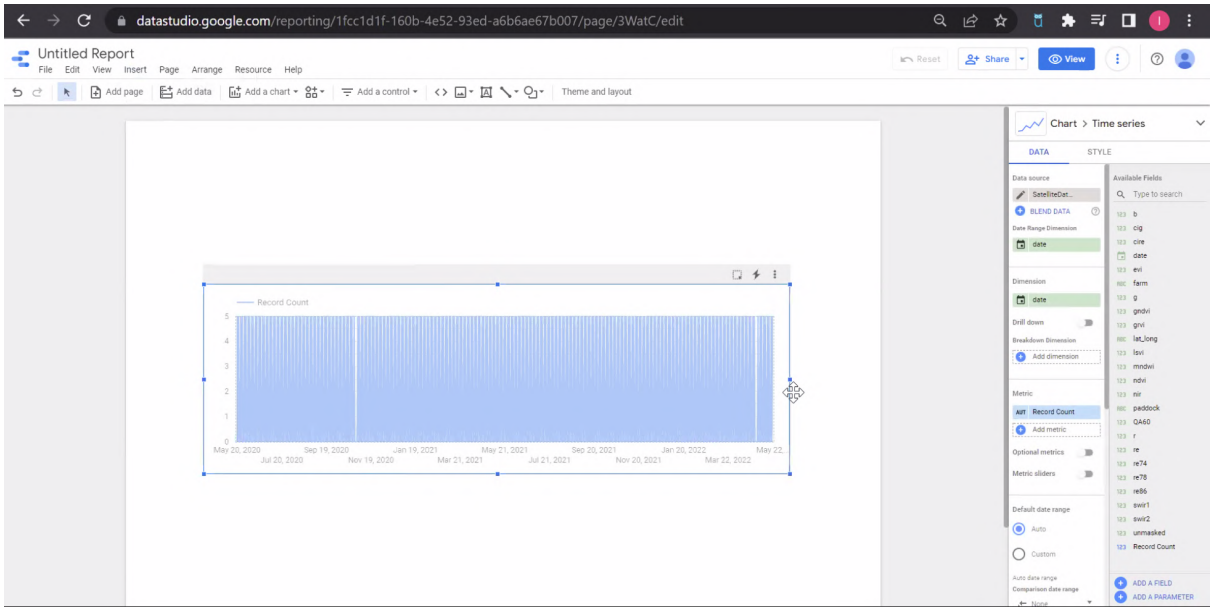
- Next, Select “add to report”:



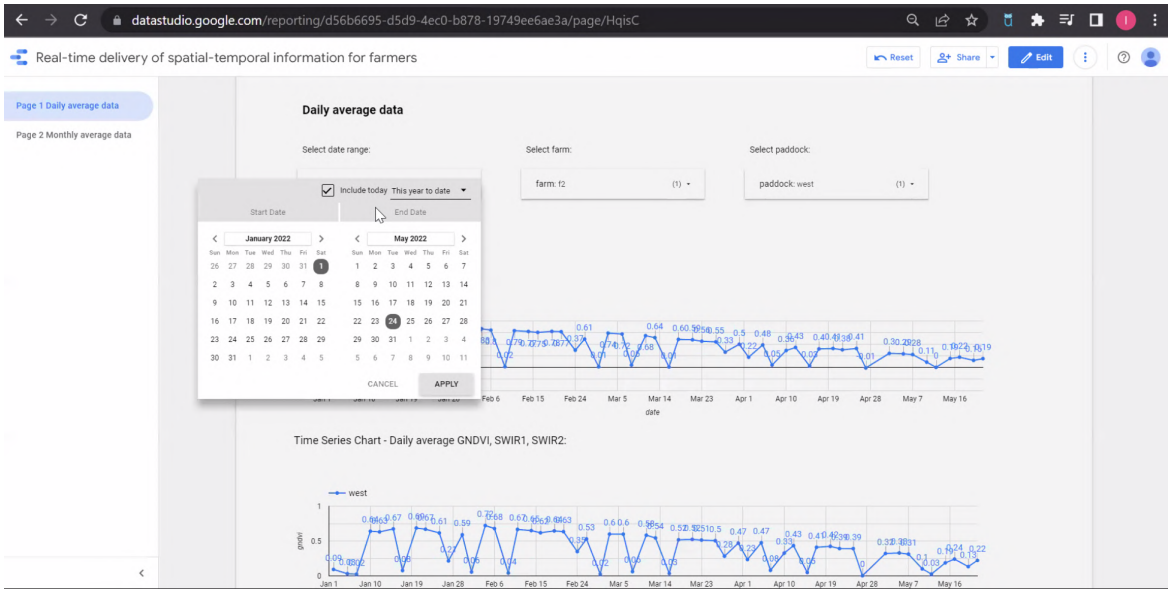
- The user has the option to pick different kinds of visual data representation formats (graphs, bubble map, etc) and can add the relevant fields of interest into the data representations within the report:



- The fields to add are located in the list to the right of the drop down screen of the selected data representation:

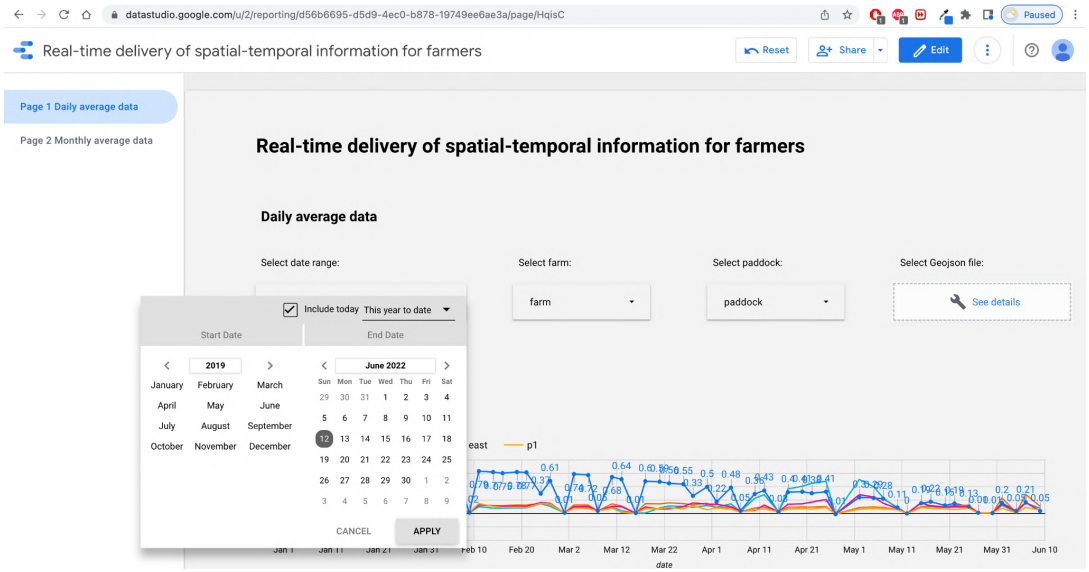


- Users can select the date range, farm or paddock from which they want to visualize data:

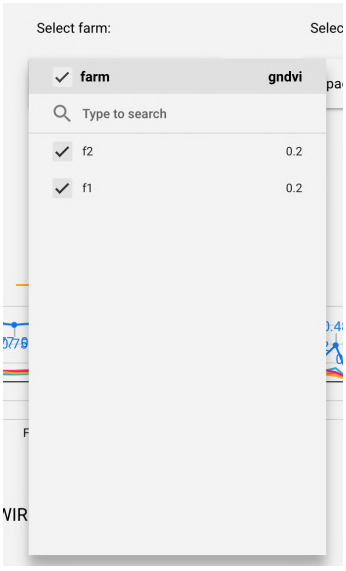


In the finished sample report the options to select are shown below:

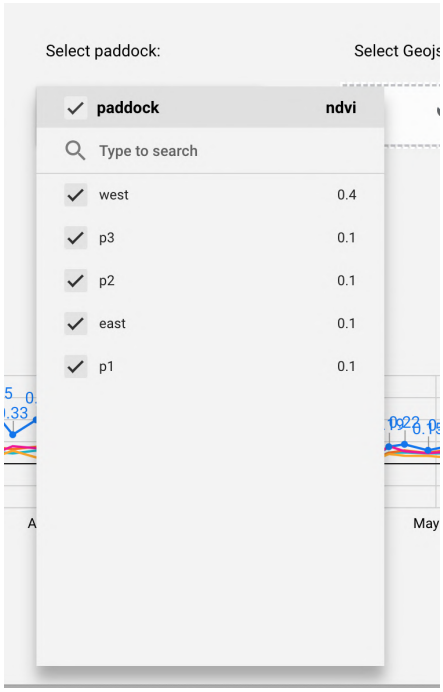
Select Date range:



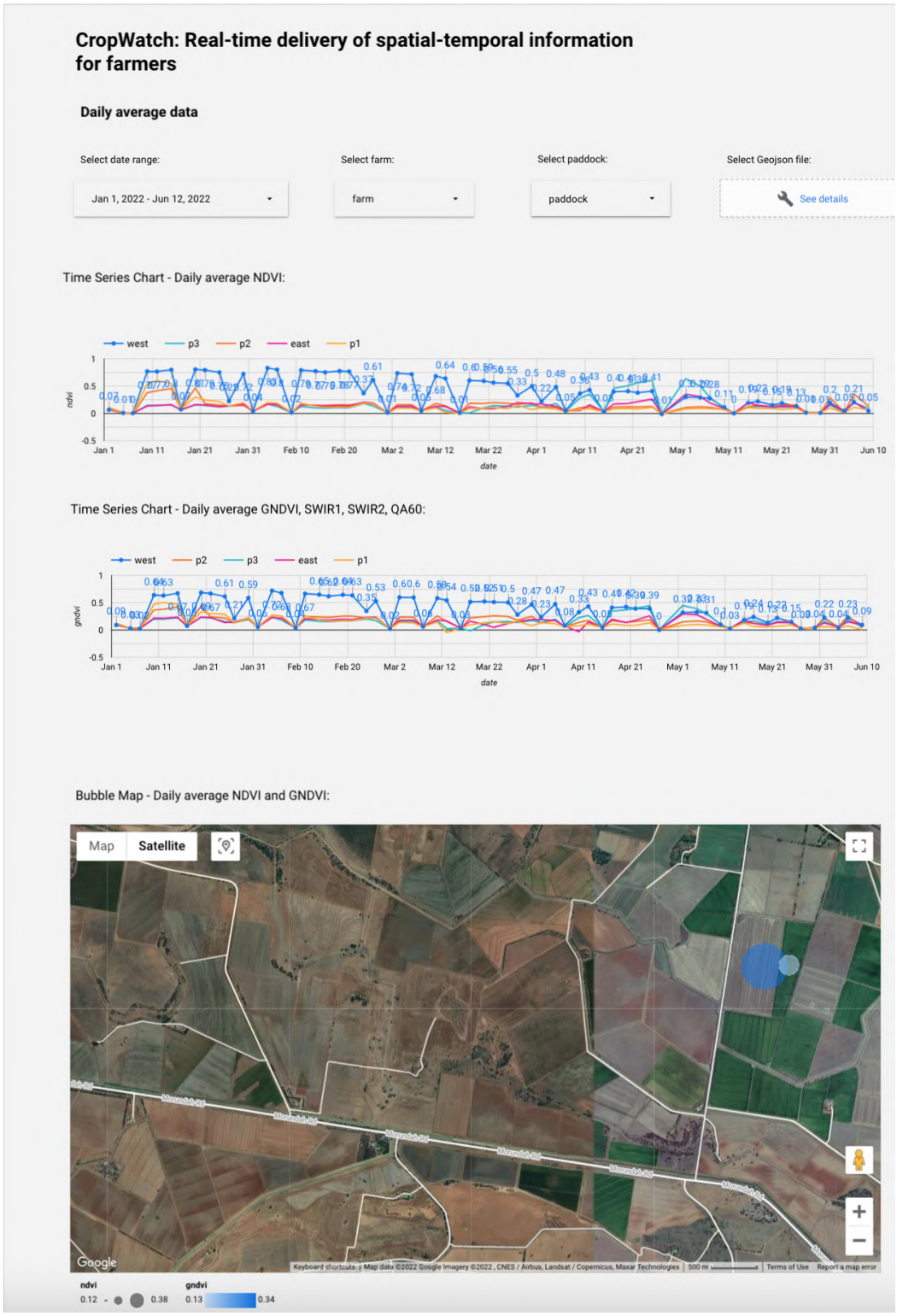
- Select farm:



- Select Paddock:

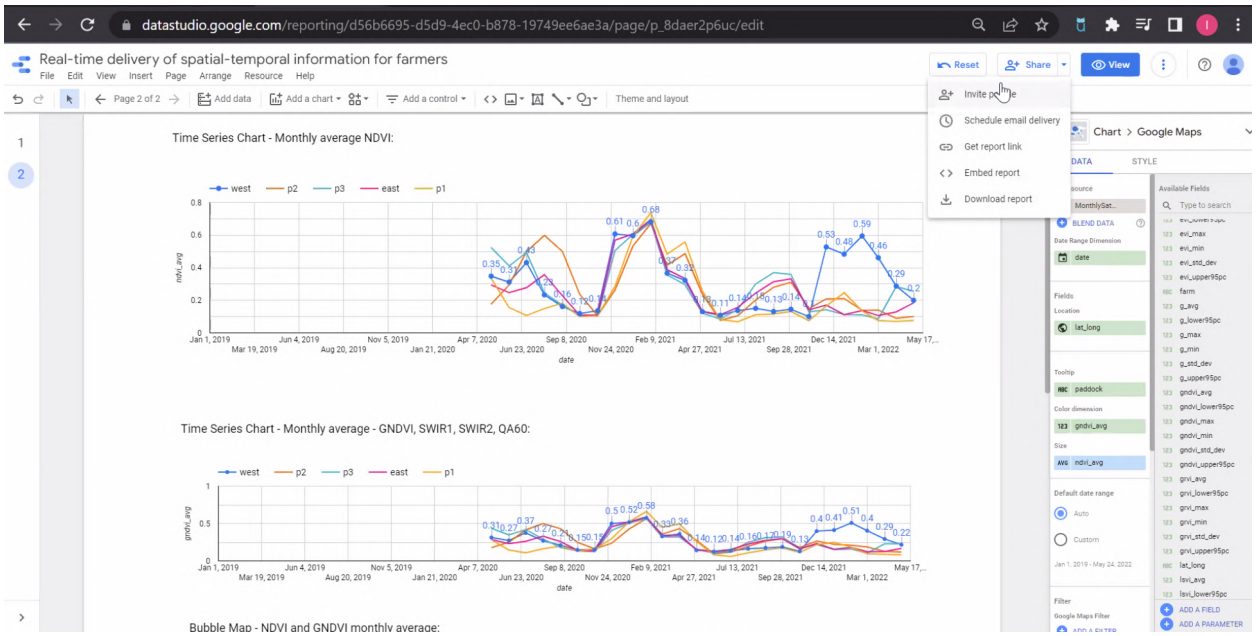


- The finished sample report is displayed below, the first page shows the “Daily average data” :

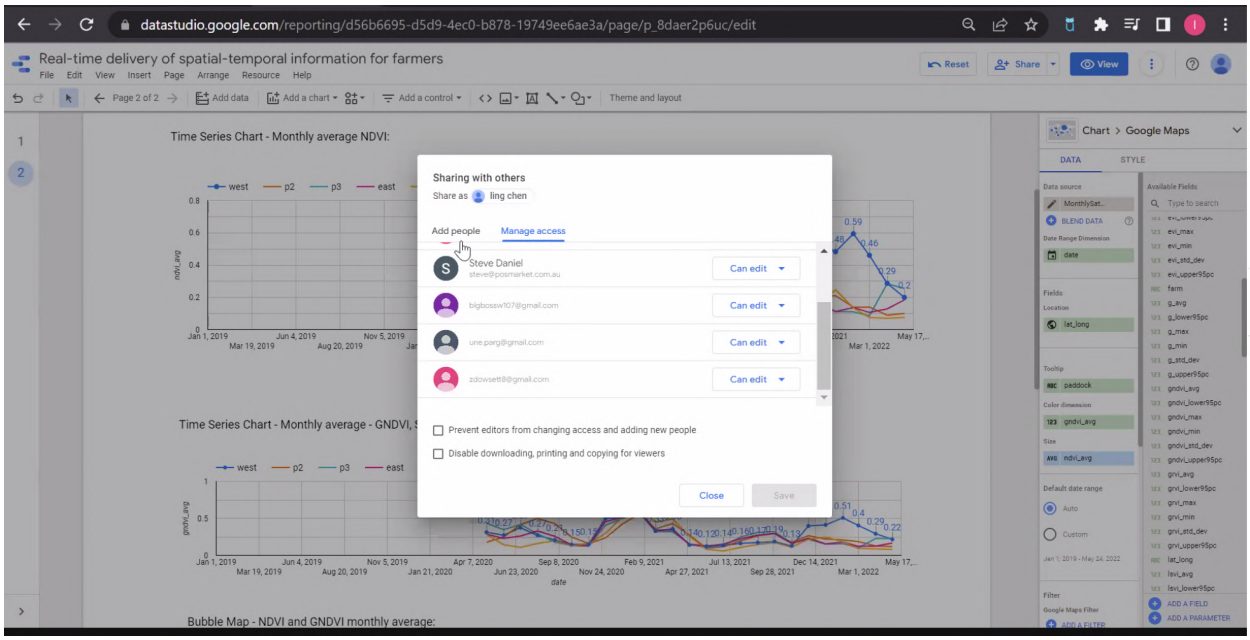


- while the second page shows the “Monthly summary data” :

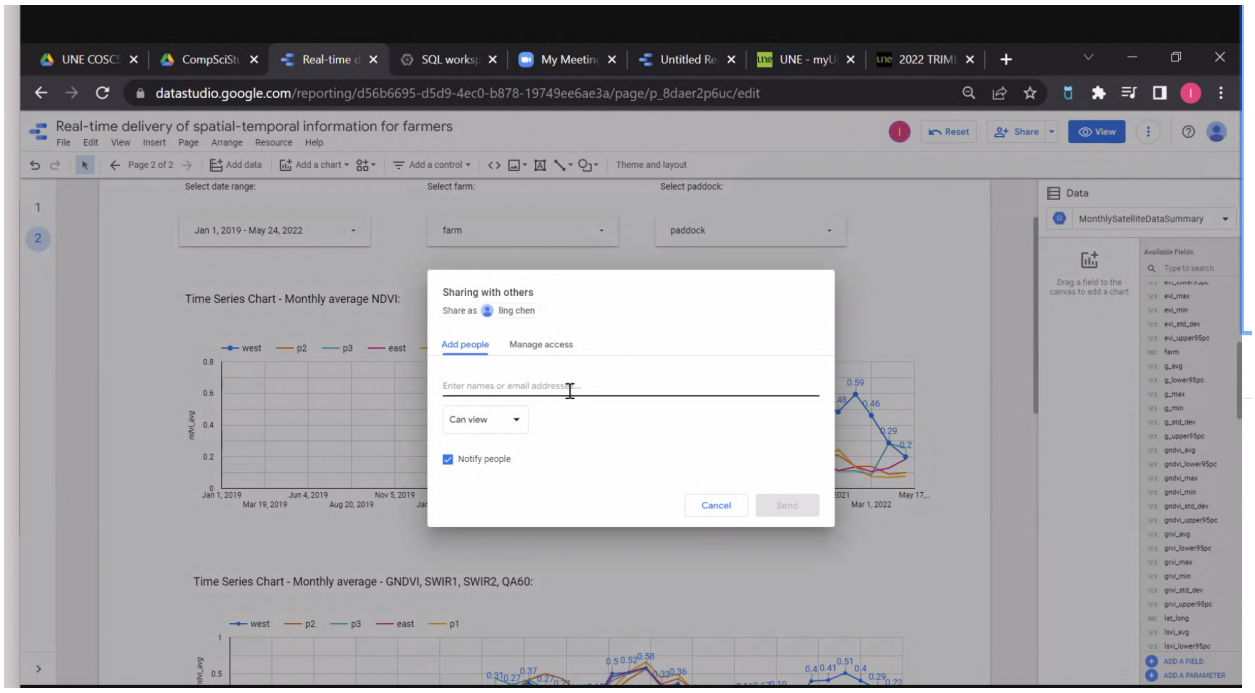
- To share the report, navigate to the top right of the page and click on the “Share+” tab button:



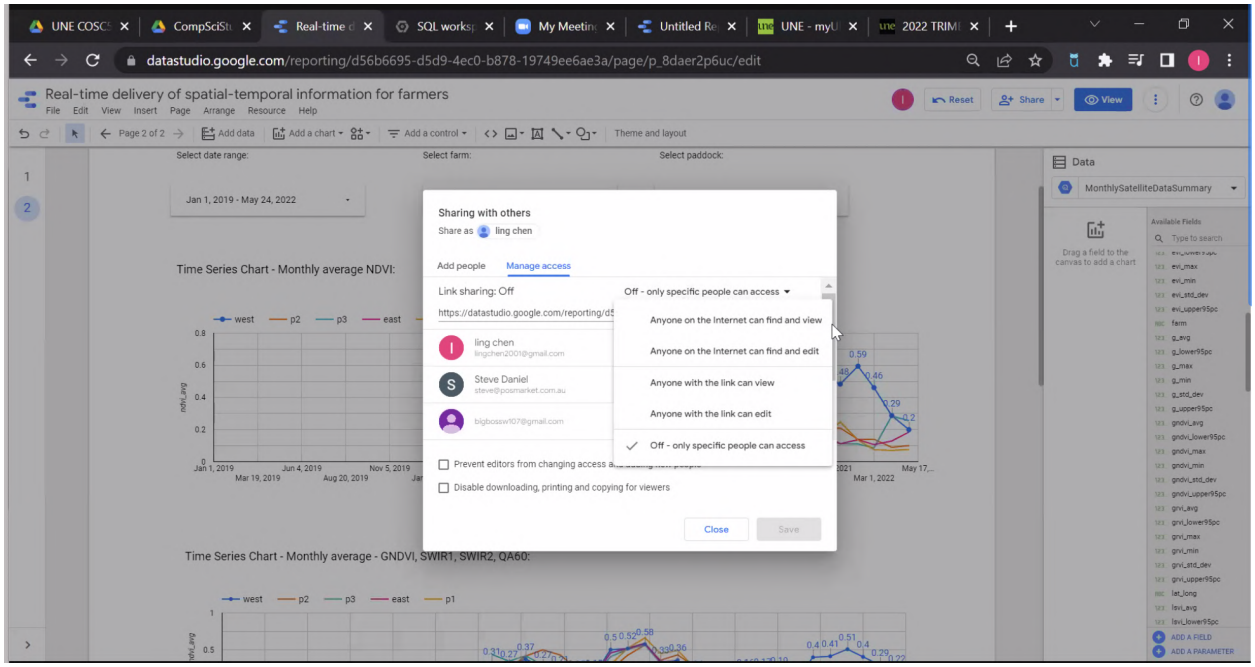
- The user can share the report by sending out an invitation for people to view the report:



- User has the option to add people to share with:



- The authorised user or owner of the report, can manage the access privileges that determine the permissions to view the report and the permissions to edit the report:



- The User is also able to generate a sharable link with the desired permissions to share publicly or with specific people, based on the intended scope of privacy:

- **Please note that:** all sharing, permissions, and management of access to the report can be accomplished via the “Share +” tab drop-down button located at the top right of the page.

This concludes the Installation and User Manual for CropWatch-RDSIF.
To visit the open source github repository established by the author of this Installation and User Manual document, go to:
https://github.com/Jase-The-Ace/CropWatch-RDSIF

NOTICES

LICENSES

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work

(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their

Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

(a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

(b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
   the Work or Derivative Works thereof, You may choose to offer,
   and charge a fee for, acceptance of support, warranty, indemnity,
   or other liability obligations and/or rights consistent with this
   License. However, in accepting such obligations, You may act only
   on Your own behalf and on Your sole responsibility, not on behalf
   of any other Contributor, and only if You agree to indemnify,
   defend, and hold each Contributor harmless for any liability
   incurred by, or claims asserted against, such Contributor by reason
   of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

   To apply the Apache License to your work, attach the following
   boilerplate notice, with the fields enclosed by brackets "[]"
   replaced with your own identifying information. (Don't include
   the brackets!) The text should be enclosed in the appropriate
   comment syntax for the file format. We also recommend that a
   file or class name and description of purpose be included on the
   same "printed page" as the copyright notice for easier
   identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.